Generating 3D Adversarial Point Clouds

Chong Xiang
Shanghai Jiao Tong University
Shanghai, China
xiangchong97@gmail.com

Charles R. Qi
Facebook AI Research
California, USA
charlesq34@gmail.com

Bo Li
University of Illinois at Urbana-Champaign
Illinois, USA
lxbosky@gmail.com

Abstract

Deep neural networks are known to be vulnerable to adversarial examples which are carefully crafted instances to cause the models to make wrong predictions. While adversarial examples for 2D images and CNNs have been extensively studied, less attention has been paid to 3D data such as point clouds. Given many safety-critical 3D applications such as autonomous driving, it is important to study how adversarial point clouds could affect current deep 3D models. In this work, we propose several novel algorithms to craft adversarial point clouds against PointNet, a widely used deep neural network for point cloud processing. Our algorithms work in two ways: adversarial point perturbation and adversarial point generation. For point perturbation, we shift existing points negligibly. For point generation, we generate either a set of independent and scattered points or a small number (1-3) of point clusters with meaningful shapes such as balls and airplanes which could be hidden in the human psyche. In addition, we formulate six perturbation measurement metrics tailored to the attacks in point clouds and conduct extensive experiments to evaluate the proposed algorithms on the ModelNet40 3D shape classification dataset. Overall, our attack algorithms achieve a success rate higher than 99% for all targeted attacks [1].

1. Introduction

Despite of the great success in various learning tasks, deep neural networks (DNNs) have been found vulnerable to adversarial examples. The adversary is able to add imperceivable perturbation to the original data and mislead DNNs with high confidence. Many algorithms have been proposed to generate adversarial examples for data such as 2D images [25, 9, 17, 15, 3], natural languages [10, 33], and audios [4, 5]. Several recent works [1, 7] have proposed adversarial examples in the 3D space, but they simply project 3D objects to 2D images as data pre-processing. However, no existing work has explored the vulnerability of actual 3D models. In this paper, we study the robustness of 3D models which directly deal with 3D objects. Specifically, we choose to represent 3D objects with *point clouds*, which are the raw data from most 3D sensors such as depth cameras and Lidars. Therefore, we attack 3D models by generating 3D adversarial point clouds.

[1]Untargeted attacks are easier to achieve with the proposed methods, so in this paper we only focus on targeted attacks.

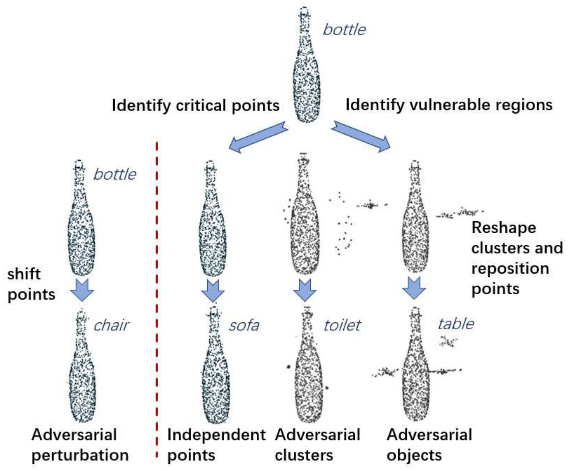

Figure 1: Attack pipeline. Our algorithms create adversarial examples by either adversarial point perturbation (left) or adversarial point generation (right). The bottle is misclassified after our attacks.

As to the attacking target, we focus on the commonly used PointNet model [19]. We choose PointNet because the model and its variants have been widely and successfully adopted in many applications such as 3D object detection for autonomous driving [18, 34, 30], semantic segmentation for indoor scene understanding [12, 19], and AI-assisted shape design [24]. Furthermore, the model has been shown to be robust to various input point perturbations and corruptions [19]. The demonstrated more robustness than 3D CNNs makes it a challenging and solid benchmark model for our evaluations. Although we focus on attacking Point-

Net, we expect our attacking algorithms and evaluation metrics extensible to more 3D models.

As the input to PointNet, a point cloud is a 3D geometric data structure that has the advantages of simple representation and low storage requirement. However, it is challenging to generate adversarial point clouds given its special properties. The point cloud's irregular format has made existing attack algorithms designed for 2D images unsuitable: 1) In raw point clouds with XYZ, there are no "pixel values" positioned in a regular structure that can be slightly modified; 2) The search space for generating new adversarial points is very large, as points can be added to arbitrary positions; 3) The commonly used L_p norm measurement in 2D images to bound perturbations does not fit for point cloud data with irregularity and varying cardinality.

To the best of our knowledge, we are the first to extend adversarial attack research to the irregular point cloud data, by addressing aforementioned challenges. We propose several novel attack methods for mainly two types of adversarial attacks on point clouds: *adversarial point perturbation* and *adversarial point generation* which are unnoticeable to human or hidden in the human psyche. The attack pipeline is illustrated in Figure 1.

For adversarial point perturbation, we propose to shift existing points negligibly. We optimize the perturbation vector under the commonly used L_p norm constraint. Our experiments show that we are able to craft unnoticeable adversarial point clouds with 100% success rate given an acceptable perturbation budget.

For adversarial point generation, We propose to synthesize and place a set of independent points or a limited number of point clusters close to the original object. In particular, we search for "vulnerable" regions of objects and optimize the positions of points and the shapes of the clusters. In total, we have three kinds of generated points, namely *independent points*, *adversarial clusters* (points in generic shapes such as balls), and *adversarial objects* (points in object shapes such as mini airplanes), shown in the right three columns in Figure 1. We constrain the generation by their sizes, their distances to the object surface as well as how many shapes we place.

Our attacks achieve 100% success rate for scattered adversarial points, 99.3% for adding three adversarial shapes, 98.2% for two, and 78.8% for one.

Furthermore, in Section 6.4 we discuss the transferrability of our 3D adversarial point clouds as well as the possibility to combine PointNet with CNNs to defense attacks in images. Sample code and data is available at `https://github.com/xiangchong1/3d-adv-pc` to support further research.

To summarize, the contributions of this paper can be summarized as follows:

- We are the first to generate 3D adversarial examples against 3D learning models and provide baseline evaluations for future research. Specifically, we choose representative point cloud data and PointNet model for our evaluations.
- We demonstrate the unique challenges in dealing with irregular data structures such as point clouds and propose novel algorithms for both *adversarial point perturbation* and *adversarial point generation*.
- We propose six different perturbation metrics tailored to different attack tasks and perform extensive experiments to show our attack algorithms can achieve a success rate higher than 99% for all targeted attacks.
- We provide robustness analysis for 3D point cloud models and show that analyzing properties of different 3D models sheds light on potential defenses for 2D instances.

2. Related Work

Point Clouds and PointNet. Point clouds are consisted of unordered points with varying cardinality, which makes it hard to be consumed by neural networks. Qi et al. [19] addressed this problem by proposing a new network called PointNet, which is now widely used for deep point cloud processing. PointNet and its variants [20, 26] exploit a single symmetric function, *max pooling*, to reduce the unordered and varying length input to a fixed-length global feature vector and thus enables end-to-end learning. [19] also tried to demonstrate the robustness of the proposed PointNet and introduced the concept of critical points and upper bounds. They showed that points sets laying between critical points and upper bounds yield the same global features and thus PointNet is robust to missing points and random perturbation. However, they did not study the robustness of PointNet against adversarial manipulations, which is the main focus of this paper.

Adversarial Examples. Szegedy et al. [25] first pointed out that machine learning models such as neural networks were vulnerable to carefully crafted adversarial perturbation. An adversarial example which appears similar to its original data can easily fool the neural networks with high confidence. Such vulnerability of machine learning models has raised great concerns in the community and many works have been proposed to improve the attack performance [9, 17, 15, 3, 16, 29, 28] and search for possible defense [2, 14, 31, 21, 22, 32]. The state-of-the-art attack algorithm, optimization based attack [3], defines an objective loss function which measures both attack effectiveness and perturbation magnitude, and uses optimization to find a near-optimal adversarial solution. However, the algorithm only deals with 2D data. Several recent works [11, 1, 7] also study the adversarial examples in the physical world. However, these works only project physical objects to 2D images and do not study models which directly deal with 3D objects. To the best of our knowledge, we are the first to generate adversarial examples for 3D machine learning models.

3. Problem Formulation

Point Cloud Data. A point cloud is a set of points which are sampled from object surfaces. A data record $x \in \mathbb{R}^{n \times 3}$ corresponds to a point set of size n, where each point is represented by a 3-tuple (x, y, z) coordinate. One most important characteristic of point cloud data is its irregularity (a point cloud is not defined on a regular grid structure), which makes it hard to adapt existing attacking algorithms from 2D images. Moreover, we are able to *add points* at any positions in the 3D space while we cannot add pixels in 2D images. However, such lack of constrain results in an extremely large search space for generative adversarial examples. New attack algorithms should be proposed to address the above problems.

Targeted Adversarial Attacks. In this paper, we only focus on targeted attacks against 3D point cloud classification models. It is flexible to extend our algorithms to other tasks like attacking segmentation models.

The goal of targeted attacks is to mislead a 3D deep model (e.g., PointNet) to classify an adversarial example as a selected target class. Formally, for a classification model $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Y}$, which maps an input $x \in \mathcal{X} \subset \mathbb{R}^{n \times 3}$ to its corresponding class label $y \in \mathcal{Y} \subset \mathbb{Z}$, an adversary has a malicious target class $t' \in \mathcal{Y}$. Based on a perturbation metric $\mathcal{D} : \mathbb{R}^{n \times 3} \times \mathbb{R}^{n' \times 3} \rightarrow \mathbb{R}$, the goal of the attack is to find a legitimate input $x' \subset \mathbb{R}^{n' \times 3}$ which:

$$\min \mathcal{D}(x, x'), \qquad s.t.\ \mathcal{F}(x') = t' \tag{1}$$

Note that, for point cloud data, n does not necessarily equal to n'.

As mentioned in [3], directly solving this problem is difficult. Therefore, we reformulate the problem as gradient-based optimization algorithms:

(?)

$$\min f(x') + \lambda * \mathcal{D}(x, x') \tag{2}$$

Here $f(x') = (\max_{i \neq t'}(\mathcal{Z}(x')_i) - \mathcal{Z}(x')_{t'})^+$ is the adversarial loss function whose output measures the possibility of a successful attack, where $\mathcal{Z}(x)_i$ is the i^{th} element of the logits (the input of softmax layer) and $(r)^+$ represents $\max(r, 0)$. By optimizing over Equation 2, we aim to search for adversarial examples with least 3D perturbation.

Attacking Types. In this paper, we consider two different types of attacks in point clouds [2]: adversarial point perturbation and adversarial point generation. In perturbation attacks, we modify existing points by shifting their XYZ positions with adversarial jitters such that a point $x_i \in \mathbb{R}^3$ in the point cloud x becomes $x'_i = x_i + \delta_i$, for $i = 1, ..., n$ where $\delta_i \in \mathbb{R}^3$ is the perturbation to the i-th point. In generation attacks, we generate a set of adversarial points $z = \{z_i | i = 1, ..., k\}$ (or $z \in \mathbb{R}^{k \times 3}$ as an array representation of it) where each $z_i \in \mathbb{R}^3$ is a new point in addition to the existing point cloud x. Then the union of the original points and adversarial points are input to the model: $x' = x \cup z$, or in the array representation $x' \in \mathbb{R}^{(n+k) \times 3}$ through array concatenation (thus $n' = n + k$). This manner of attacking is very new and vastly different from attacks in images, because we cannot generate new pixels in a fix-sized image.

[2]To guarantee the points can still cover the object surface, we do not allow an adversary to remove points.

4. Adversarial Point Perturbation

In this section, we focus on the first and the simpler type of point cloud attack: adversarial point perturbation. Since for perturbation we have correspondences between the original points and the perturbed ones, we can simply use L_p norm to measure the distance between the two clouds.

L_p **Norm.** The L_p norm is a commonly used metric for adversarial perturbation of fixed-shape data. For the original point sets $\mathcal{S}$ and corresponding adversarial set $\mathcal{S}'$, the L_p norm of the perturbation is defined as:

$$\mathcal{D}_{L_p}(\mathcal{S}, \mathcal{S}') = (\sum_i (s_i - s'_i)^p)^{\frac{1}{p}} \tag{3}$$

where s_i is the i^{th} point coordinate in set $\mathcal{S}$, and s'_i is its corresponding point in set $\mathcal{S}'$.

We can directly use Equation 2 to generate the adversarial perturbations $\{\delta_i\}_{i=1}^n$, by optimization with the L_2 norm distance to bound the perturbation.

5. Adversarial Point Generation

Besides perturbing existing points, another general type of attacking strategy is to generate new adversarial points to mislead the 3D model. Among the ways to generate new points, a simple approach is to add arbitrary number of *independent points* (Section 5.1), ideally close to the object surface so that they are unnoticeable [3].

On the other hand, we consider a more challenging attack task where the adversary is only able to add a limited number (1-3) of adversarial shapes (Section 5.2 and Section 5.3), as either generic primitive shapes such as balls or meaningful shapes such as small airplane models. The task is challenging since points can only be added within small regions of the 3D space and the points of original object remain unchanged. The goal of this attack is to generate

[3]How to realize this attack in real world is still a question though.

adversarial point clusters that are plausible so they cloud be hidden in the human psyche.

In the following subsections, we will introduce metrics and our attacking algorithms to generate adversarial individual points, as well as two kinds of adversarial shapes: *adversarial clusters* and *adversarial objects*.

5.1. Generating Adversarial Independent Points

In this section, we focus on the attack of generating (unnoticeable) independent points. Note that when adding new points to the original point clouds, we have to deal with data dimensionality changes. We first introduce metrics that measure the deviation of adversarial points to the original one and then desrcribe our attack algorithm.

5.1.1 Perturbation Metrics

Hausdorff Distance. Hausdorff distance is often used to measure how far two subsets of a metric space are from each other. Formally, for an original point set $\mathcal{S}$ and its adversarial counterpart $\mathcal{S}'$, we define Hausdorff distance as:

max in the min dis

$$\mathcal{D}_H(\mathcal{S}, \mathcal{S}') = \max_{y \in \mathcal{S}'} \min_{x \in \mathcal{S}} \|x - y\|_2^2 \tag{4}$$

Intuitively, Hausdorff distance finds the nearest original point for each adversarial point and outputs the maximum square distance among all such nearest point pairs. We do not include the term $\max_{x \in \mathcal{S}} \min_{y \in \mathcal{S}'} \|x - y\|_2^2$ since we do not modify the original object $\mathcal{S}$.

Chamfer Measurement.[4] Chamfer measurement [8] is a similar perturbation metric as Hausdorff distance. The difference is Chamfer Measurement takes the average, rather than the maximum, of the distances of all nearest point pairs. The formal definition is as follows:

$$\mathcal{D}_C(\mathcal{S}, \mathcal{S}') = \frac{1}{\|\mathcal{S}'\|_0} \sum_{y \in \mathcal{S}'} \min_{x \in \mathcal{S}} \|x - y\|_2^2 \tag{5}$$

Number of Points Added. We also want to measure the number of points added in our attack, by counting points whose distances from the object surface is above a certain threshold. Formally, for an original point set $\mathcal{S}$, the generated point set $\mathcal{S}'$, and a threshold value T_{thre}, the number of points added is defined as:

$$Count(\mathcal{S}, \mathcal{S}') = \sum_{y \in \mathcal{S}'} \mathbb{1}[\min_{x \in \mathcal{S}} \|x - y\|_2 > T_{thre}] \tag{6}$$

where $\mathbb{1}[\cdot]$ is the indicator function whose value is 1 when the statement is true and 0 otherwise. Note that the number of points added is not optimized as the perturbation metric $\mathcal{D}$ in Equation 2 due to its incompatibility with gradient-based optimization algorithms, but is reported as an additional performance metric.

[4] We name it as "Chamfer measurement" since this perturbation metric does not satisfy triangle inequality, which means it does not satisfy the definition of distance.

5.1.2 Attacking Algorithm

Directly adding points to the unconstrained 3D space is infeasible due to the large search space. Therefore we propose an *initialize-and-shift* method to find appropriate position for each added point:

1. *Initialize* a number of points to the same coordinates of existing points as initial points.
2. *Shift* initial points via optimizing Equation 2 and output their final positions.

During the optimization process, some initial points are shifted from their initial positions and "added" to the original objects as adversarial points. The others that are barely shifted do not change the shape of the object, and thus can be discarded as points-not-added.

To make the optimization more efficient, we propose to initialize points to the positions of "critical points" of the target. Critical points are like key points or salient points in a 3D point cloud. In PointNet specifically, they can be computed by taking the points that remain active after the max pooling [19], which means they are at important positions that determine the object category. Adversarial points around these critical positions are more likely to change the final prediction.

$\rightarrow$ 关键特征.

We use Hausdorff and Chamfer measurements as the perturbation metrics $\mathcal{D}$ for this attack because they are more capable of measuring how unnoticeable the adversarial point clouds of different dimensionality are.

5.2. Generating Adversarial Clusters

For *adversarial clusters*, we aim to minimize the radius of the generated cluster so that they look like a ball attached to the original object and will not arouse suspicion. In addition, we also encourage the cluster to be close to the object surface. To satisfy these two requirements, we introduce the perturbation metrics used as follows.

5.2.1 Perturbation Metrics

Farthest Distance. If the farthest pair-wise point distance in a point set is controlled within a certain threshold, the points in this set are able to form a shaped cluster. Formally, we define farthest distance of a point set $\mathcal{S}$ as:

$$\mathcal{D}_{far}(\mathcal{S}) = \max_{x, y \in \mathcal{S}} \|x - y\|_2 \tag{7}$$

Chamfer Measurement. Besides encouraging point cluster to form within a small radius, we may also want to push the added clusters towards the surface of the object. Therefore, we also include the Chamfer Measurement (defined in Equation 5) as our perturbation metric and optimization objective.

Number of Clusters Added. Similar to the number of points added in the unnoticeable adversarial point cloud generation, the number of clusters added also serves as an additional metric for attack performance, which is hard bounded to 1-3 in our experiments.

5.2.2 Attacking Algorithm

Before going into the details of generation algorithms, we need to reformulate Equation 2 as follow:

$$\min f(x') + \lambda \cdot (\sum_i \mathcal{D}_{far}(\mathcal{S}_i) + \mu \cdot \mathcal{D}_C(\mathcal{S}_0, \mathcal{S}_i)) \tag{8}$$

where $i \in \{1, 2, \dots, m\}$, $\mathcal{S}_0$ is the original object, $\mathcal{S}_i$ is the i^{th} adversarial point cluster, m is number of adversarial clusters, and μ is the weight used to balance the importance between Farthest Distance loss and Chamfer Measurement loss. Here we abuse the notation a little to use $\mathcal{D}$ to denote both mappings $\mathbb{R}^{n\times 3} \times \mathbb{R}^{n'\times 3} \to \mathbb{R}$ and $\mathbb{R}^{n\times 3} \to \mathbb{R}$.

Generating adversarial clusters is a special case of adding adversarial point clouds, so we can adopt the *initialize-and-shift* method used. However, unlike independent point generation, we have to constrain the added points clustered to be within small regions. As points are likely to get stuck in their initialized vicinity due to the ubiquity of local-minima, we need a more efficient initialization methods for adversarial clusters generation.

We try to leverage the idea of "vulnerable regions" for initialization. For formatted data like 2D images, it is common to impose a L_1 constraint to encourage the sparsity of the perturbation vector. The region with large perturbation under a proper L_1 constraint is believed to be important for model decisions and thus vulnerable to adversarial attacks. However, the L_1 constraint is not well defined on point clouds thus inapplicable here. Instead, we take advantage of "critical points" again to effectively find potentially vulnerable regions for initialization. Critical points, as a subset of the original set, collectively determine the global features of the object shape but could also be vulnerable regions to attacks.

Given a victim object and a target class t', the attack process is as follows:

1. Obtain the critical points of the objects in *target* class.
2. Use the clustering algorithm DBSCAN [6] to cluster the selected critical points.
3. Choose points in the k largest clusters as the initial points, where k is a self-chosen parameter as well as a metric for attack performance evaluation.
4. Optimize over Equation 8 using gradient-based algorithms and find optimal cluster positions and shapes.

Note that DBSCAN groups points that are closely packed (or points with local density passing a threshold), while marking the other points lying in low-density regions as outliers [6]. Thus, we are able to filter out outlier points and get compact clusters via it.

Besides tuning DBSCAN, it is essential to determine the number of objects in target class we use, as well as the number of critical points selected from each target object. Choosing only one target object restricts the space distributions of the critical points. However, using too many target objects result in density scattered critical points unhelpful to identify a sparse set of vulnerable regions. The reasons for tuning the number of critical points selected are similar, we want a moderate number of them. However, such parameter tuning does not need to too fine grained as the attack pipeline is still dominated by the optimization over Equation 8.

5.3. Generating Adversarial Objects

For this attack, we start from some meaningful objects like small airplanes, slightly modify them, and place them in the appropriate adversarial positions. People may not become suspicious because the adversarial objects are like other benign objects nearby.

5.3.1 Perturbation Metrics

L_p **Norm.** Since we want to only slightly modify the meaningful objects and make the generated shapes similar to the real-world ones, we adopt the L_p, specifically L_2, as our first metric.

Chamfer Measurement. Similar to the adversarial clusters, we want to encourage the generate shape to be close the original object.

Number of Clusters Added. Number of clusters added, bounded to 1-3, is also used to evaluate the attack performance.

5.3.2 Attacking Algorithm

We also need to rewrite the objective function to fit the attack setting:

$$\min f(x') + \lambda \cdot (\sum_i \mathcal{D}_{L_2}(\mathcal{S}_{i0}, \mathcal{S}_i) + \mu \cdot \mathcal{D}_C(\mathcal{S}_0, \mathcal{S}_i)) \tag{9}$$

where $i \in \{1, 2, \dots, m\}$, $\mathcal{S}_0$ is the original object, $\mathcal{S}_i$ is the i^{th} adversarial point cluster, $\mathcal{S}_{i0}$ is the i^{th} real-world clusters, m is number of adversarial clusters, and μ is the weight used to balance the importance between L_2 loss and Hausdorff Distance loss. To mount this attack, we need to find the vulnerable regions first and then to initialize the perturb the added real-world point clusters. The attack pipeline is as follows:

1. Obtain the critical points of the objects in *target* class.
2. Use the clustering algorithm DBSCAN [6] to cluster the selected critical points.
3. Identify the k largest clusters and calculate the position of cluster centers, where k is a self-chosen parameter as well as a metric for attack performance evaluation.
4. Choose meaningful objects and initialize them to make their centers overlap with the calculated positions in the previous step.
5. Optimize over Equation 9 using gradient-based algorithms and find optimal cluster positions and shapes.

Note that we also have the freedom to choose the orientation of the modified clusters. Since adversarial clusters with different orientations would not arouse suspicion, we do not impose an constraint on the magnitude of rotation.

6. Experiment Results

In this section, we implement the proposed algorithms for different attack tasks and conduct extensively evaluation on attack performance based on various metrics.

6.1. Dataset and 3D Models

We use the aligned benchmark ModelNet40 [27, 23] dataset for our experiments. The ModelNet40 dataset contains 12,311 CAD models from 40 most common object categories in the world. 9,843 objects are used for training and the other 2,468 for testing. As done by Qi et al. [19], we uniformly sample 1,024 points from the surface of each object, and re-scale them into a unit ball. We use the same PointNet structure as proposed in [19] and train the model with all ModelNet40 training data to obtain our victim model. The ModelNet40 dataset is very imbalanced. For our attacks, we randomly select 25 test examples from each 10 largest classes, namely airplane, bed, bookshelf, bottle, chair, monitor, sofa, table, toilet and vase, to generated adversarial point clouds for. For each victim data record, we generate adversarial examples targeted on the rest 9 classes. Therefore, we have 2,250 (victim,target) attach pairs for our experiments.

6.2. Adversarial Point Perturbation Evaluation

We evaluate the attack performance for adversarial point perturbation in this subsection. We use L_2 distance as the perturbation constraint $\mathcal{D}$ in Equation 2 and minimize the objective loss to find the optimal perturbation. To obtain good attack performance, it is essential to choose an appropriate value for the weight λ, which controls the balance between minimizing adversarial loss and perturbation magnitude. If the λ is too small, the perturbation constraint is

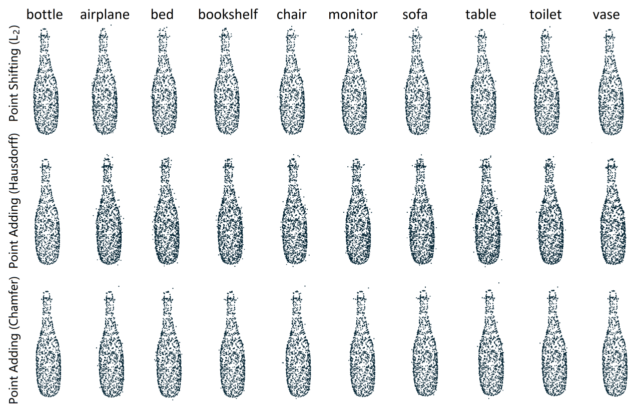

Figure 2: Visualization for adversarial point perturbation.

not strong enough and the perturbation would become too obvious. On the other hand, a λ that is too large would result in minimizing perturbation magnitude only and fail to attack. For all of our attacks, we perform 10-step binary search for the near-optimal λ. During the search, we record the smallest perturbation $\mathcal{D}(x, x')$ and its corresponding adversarial example x', and finally output the most unnoticeable adversarial example.

we report the experiment results for three cases: *best case* for the most easily attacked (victim,target) class pair, *average case* for all attacking class pairs, and *worst case* for the most difficult pair. The success rate and mean perturbation loss for point shifting attacks are reported in the first two columns of Table 1. We can see we successfully attack all victims examples into all target classes. The perturbation loss for this attack is relatively small, considering the perturbation vector contains 1,024 elements. We also provide visualization in the first row of Figure 2. We choose class "bottle" as our visualization victim because adversarial perturbation would become more obvious for a simple shape like a bottle. More visualization for other objects can be found in the supplementary. From the visualization, we can see the perturbation (the adversarial point cloud) is nearly indistinguishable.

6.3. Adversarial Point Generation Evaluation

In this subsection, we evaluate the attack performance of three different ways of adversarial point generation: *independent points*, *adversarial clusters*, and *adversarial objects*.

Adversarial Independent Points. For generating adversarial independent points, we take Hausdorff and Chamfer measurements as perturbation metrics $\mathcal{D}$ and optimize over Equation 2. We use two different distances separately and compare performance of these two constraints. To calculate the number of points added, we get the critical points of the newly generated adversarial point clouds, set T_{thre}

Case	Shifting Points (L_2 Norm)		Adding Points ($\mathcal{D}_H$)			Adding Points ($\mathcal{D}_C$)		
	mean loss	success rate	mean loss	#points added	success rate	mean loss	#points added	success rate
Best	0.0874	100%	0.0003	93	100%	3.1×10^{-5}	58	100%
Average	0.3032	100%	0.0105	88	100%	2.7×10^{-4}	51	100%
Worst	0.4674	100%	0.0210	99	100%	7.2×10^{-4}	49	100%

Table 1: Attack performance evaluation for adversarial point perturbation and adversarial independent point generation

Attack	#Shape 1			#Shape 2			#Shape 3		
	$\mathcal{D}_{far}$ / $\mathcal{D}_{L_2}$	$\mathcal{D}_C$	success rate	$\mathcal{D}_{far}$ / $\mathcal{D}_{L_2}$	$\mathcal{D}_C$	success rate	$\mathcal{D}_{far}$ / $\mathcal{D}_{L_2}$	$\mathcal{D}_C$	success rate
adversarial clusters	0.5401	0.1374	78.8%	0.3118	0.1839	98.2%	0.1818	0.1744	99.3%
adversarial objects	0.5539	0.1776	54.6%	0.0838	0.1332	93.8%	0.0212	0.0855	97.3%

Table 2: Attack performance evaluation for adversarial clusters and adversarial objects (average case).

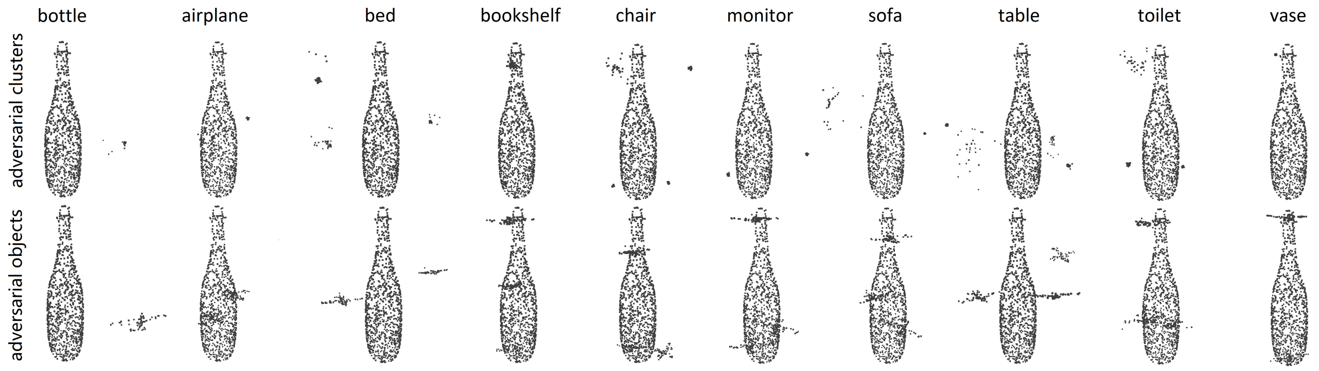

Figure 3: Visualization for adding 3 adversarial clusters/objects.

to 0.01, and count over points moved further than it. The experiment results are shown in Table 1. First, both Hausdorff and Chamfer constraints result in attack success rate of 100%, proving the effectiveness of proposed *initialize-and-shift* algorithm. Secondly, we can see great difference between the mean distance losses and number of points added. Since Hausdorff distance only controls the largest distance of all nearest point pairs while Chamfer measurement calculates the average distance, the loss value of Hausdorff is much larger than that of Chamfer and so does the number of points added. A more detailed analysis on the different attack performance of Hausdorff and Chamfer measurement is included in the supplementary.

The visualization of the adversarial bottle by adding points is shown in the second and third rows of Figure 2. From the figure, we can easily observe the different characteristics of different constraints (Hausdorff constraint results in more added points while Chamfer constraint leads to more obvious outliers). Considering the different properties of the two constraints, one can combine these two constraints and adjust the weights for the two perturbation metrics according to the specific attack goals.

Adversarial Clusters. To get initial clusters, we randomly select 8 different objects from the target class test set, and obtain 32 most important critical points for each selected object based on the number of global feature channels it contributes to. We then use DBSCAN algorithm to cluster these 32×8 critical points. After that, we retain k clusters of largest size and discard other small clusters and outliers. For each cluster, we conduct subsampling or padding to obtain 32 initial points. We vary k from 1 to 3 to see how the number of adversarial clusters affects the attack performance (success rate and distance loss). In Equation 8, the parameter λ is chosen via 5-step binary search while μ is prefixed according to the adversary's preference on smaller or closer clusters. In our experiment, we set μ to 0.1. Due to the lack of space, we only report the quantitative results for *average case* in Table 2. The comprehensive results for three cases are included in the supplementary.

The table shows that, as the number of adversarial clusters increases, the attack success rate is significantly improved and we are able to attack 99.3% examples when adding 3 adversarial clusters. Moreover, a larger number of added clusters also helps reduce the perturbation loss for each cluster. When we only add one cluster, the farthest distance of the cluster for average case is 0.5401, which is

	Shift	Add (D_H)	Add (D_C)	3 Clusters	3 Objects
PointNet++ [20]	3.9%	8.9%	3.6%	8.9%	9.6%
DGCNN [26]	1.9%	6.8%	7.4%	16.9%	16.5%
Augmented PointNet	3.0%	4.0%	5.7%	46.5%	34.5%
Different initialization	5.5%	7.4%	7.8%	48.3%	39.2%

Table 3: Attack success rates of untargeted transfer attacks against PointNet++, DGCNN, augmented PointNet, and PointNet with a different weight initialization.

ϵ	0	0.1	0.2	0.3	0.4	0.5
LeNet (original)	99.2%	70.7%	35.5%	18.7%	12.0%	9.0%
LeNet (binarized)	98.9%	97.2%	93.3%	86.2%	75.4%	28.0%
PointNet	99.0%	98.6%	98.1%	96.8%	94.1%	63.8%

Table 4: Test accuracy of adversarial examples on MNIST.

quite large considering the whole object fits in a unit ball. However, the farthest distance drops dramatically to 0.1818 when we are adding 3 clusters. Thus, it is reasonable to expect better attack performance if the adversary is able to add more than 3 clusters. Visualization for adding 3 adversarial clusters can be found in the first row of Figure 3. Several small clusters are clearly shown for most attack pairs.

Adversarial Objects. For this attack, we use the same setting as the adversarial cluster to identify the vulnerable regions. We randomly select an object from the "airplane" class and initialize it to the centers of different vulnerable regions. The airplane is re-scaled to three-tenths of its original size, and 64 points are uniformly sampled from the surface. [5] After the initialization, we optimize according to Equation 9 to perform the attack. Similarly, the parameter λ is determined by binary search while the μ is pre-set to 0.2 in our experiment.

The results for different perturbation metrics are provided in the second row of Table 2. We can this attack is more challenging than that of adversarial clusters since the shapes of the objects are almost predefined. However, the shapes similar to real-world objects made this attack less suspicious. Moreover, we still achieve a reasonably high success rate of 97.3% when adding three adversarial clusters, and we can achieve better performance if more adversarial objects are allowed to be added. We also provide visualization in the second row of Figure 3. We can see the several small airplanes near the bottle are already capable to fool the PointNet model. Comparison between two rows of visualization in Figure 3 shows that when the clusters are close to the object, *adversarial clusters* have better visualization performance while *adversarial objects* are less suspicious when the clusters are far from the surface. This further justifies our attempt to introduce two kinds of meaningful shapes.

[5]We choose an airplane to simulate the scenario where the adversary could manipulate several micro-UAVs to suspend around the victim object.

6.4. More Analysis on 3D Model Robustness

In this subsection, we provide robustness analyses for PointNet-like models.

Transferablity of Adversarial Point Clouds. We feed our crafted adversarial point clouds to PointNet++ [20], DGCNN [26], a PointNet trained with data augmentation, and a PointNet trained with a different weight initialization and find that theses 3D adversarial examples actually hardly transfer as targeted attacks. Furthermore, we calculate the success rate for untargeted attack and the results are shown Table 3. We can see the transferability for untargeted attack is also limited compared with 2D adversarial examples. Since our proposed attack methods are general and can be applied to attack other 3D models, the low transferability may be related with special properties of 3D models themselves. This intrinsic property makes it possible to design black-box defense against such adversarial instances.

Defense on MNIST [13] with PointNet. Motivated by aforementioned robustness analysis, we take a step forward to use PointNet structure for defense on MNIST dataset. We binarize the grey-scale images and sample 256 points from each MNIST digit. We craft adversarial examples by attacking LeNet [13] with FGSM [9]. The test accuracy of binarized images (LeNet) and sampled point clouds (PointNet) with different values of attack parameter ϵ are reported in Table 4. We can see the PointNet model achieve relatively high test accuracy and show promising defense properties against adversarial examples.

In summary, our analysis shows that PointNet structure is more robust than traditional CNNs. We believe part of the robustness comes from the learning of global features via max pooling. Though the intriguing properties is not fully understood yet, we believe a further study on this would motivate a defense direction to include PointNet-like structure to improve the robustness of traditional nerual networks.

7. Conclusion

Arguable as the first to study the vulnerability of 3D learning models, in this paper, we have proposed several attacking algorithms to generate adversarial point clouds to fool the widely used PointNet model, including adversarial point perturbation and adversarial point generation. We also propose six different perturbation metrics and extensively evaluate the performance of the proposed attack algorithms. Our extensive experiment results show that the proposed algorithms are able to find 3D adversarial point clouds with an attack success rate higher than 99% given an acceptable perturbation budget. We hope this work is able to provide a baseline as well as a guideline for future 3D adversarial example research.

References

[1] A. Athalye and I. Sutskever. Synthesizing robust adversarial examples. *arXiv preprint arXiv:1707.07397*, 2017. 1, 2

[2] N. Carlini and D. Wagner. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*, 2016. 2

[3] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 39–57. IEEE, 2017. 1, 2, 3

[4] N. Carlini and D. Wagner. Audio adversarial examples: Targeted attacks on speech-to-text. *arXiv preprint arXiv:1801.01944*, 2018. 1

[5] N. Das, M. Shanbhogue, S.-T. Chen, L. Chen, M. E. Kounavis, and D. H. Chau. Adagio: Interactive experimentation with adversarial attack and defense for audio. *arXiv preprint arXiv:1805.11852*, 2018. 1

[6] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231, 1996. 5, 6

[7] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song. Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945*, 1, 2017. 1, 2

[8] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, volume 2, page 6, 2017. 4

[9] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1, 2, 8

[10] R. Jia and P. Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017. 1

[11] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. 2

[12] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. *arXiv preprint arXiv:1711.09869*, 2017. 1

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 8

[14] D. Meng and H. Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147. ACM, 2017. 2

[15] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2574–2582, 2016. 1, 2

[16] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017. 2

[17] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016. 1, 2

[18] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2018. 1

[19] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016. 1, 2, 4, 6

[20] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 2, 8

[21] R. Ranjan, S. Sankaranarayanan, C. D. Castillo, and R. Chellappa. Improving network robustness against adversarial attacks with compact convolution. *arXiv preprint arXiv:1712.00699*, 2017. 2

[22] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018. 2

[23] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox. Orientation-boosted voxel nets for 3d object recognition. In *British Machine Vision Conference (BMVC)*, 2017. 6

[24] M. Sung, H. Su, V. G. Kim, S. Chaudhuri, and L. Guibas. Complementme: weakly-supervised component suggestions for 3d modeling. *ACM Transactions on Graphics (TOG)*, 36(6):226, 2017. 1

[25] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1, 2

[26] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018. 2, 8

[27] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 6

[28] C. Xiao, B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018. 2

[29] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song. Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*, 2018. 2

[30] D. Xu, D. Anguelov, and A. Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2018. 1

[31] W. Xu, D. Evans, and Y. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017. 2

[32] Z. Yan, Y. Guo, and C. Zhang. Deepdefense: Training deep neural networks with improved robustness. *arXiv preprint arXiv:1803.00404*, 2018. 2

[33] Z. Zhao, D. Dua, and S. Singh. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*, 2017. 1

[34] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2018. 1