Robust Adversarial Objects against Deep Learning Models

Tzungyu Tsai[1], **Kaichen Yang**[2], **Tsung-Yi Ho**[1], **Yier Jin**[2*]
[1]National Tsing Hua University, Hsinchu, Taiwan
[2]University of Florida, USA
s107062519@m107.nthu.edu.tw, bojanykc@ufl.edu, tyho@cs.nthu.edu.tw, yier.jin@ece.ufl.edu

Abstract

Previous work has shown that Deep Neural Networks (DNNs), including those currently in use in many fields, are extremely vulnerable to maliciously crafted inputs, known as adversarial examples. Despite extensive and thorough research of adversarial examples in many areas, adversarial 3D data, such as point clouds, remain comparatively unexplored. The study of adversarial 3D data is crucial considering its impact in real-life, high-stakes scenarios including autonomous driving. In this paper, we propose a novel adversarial attack against PointNet++, a deep neural network that performs classification and segmentation tasks using features learned directly from raw 3D points. In comparison to existing works, our attack generates not only adversarial point clouds, but also robust adversarial objects that in turn generate adversarial point clouds when sampled both in simulation and after construction in real world. We also demonstrate that our objects can bypass existing defense mechanisms designed especially against adversarial 3D data.

1 Introduction

Deep learning achieves great success in many areas such as image classification, object detection, and voice recognition. Inspired by these achievements, many innovative neural network architectures are proposed to process 3D data, particularly point cloud data. PointNet (Qi et al. 2017a) and its variation, PointNet++ (Qi et al. 2017b), are popular network models designed for classification and segmentation on point data. In contrast to existing voxel-based or image-based methods (Maturana and Scherer 2015; Qi et al. 2016) which transform point data into regular 3D voxel grids or render points to 2D images, PointNets make predictions based on the features learned directly from raw 3D points, thus utilizing more information for better accuracy.

Though deep learning is recognized as a promising way in processing different tasks, recent works (Szegedy et al. 2013) have shown that deep learning models are vulnerable to deliberately crafted inputs, known as adversarial examples. Adversarial examples are generated by adding small but intentionally selected perturbations to the original inputs that lead the target models to specific incorrect outputs. The phenomenon of adversarial examples was first found in deep learning models designed for image classification tasks (Szegedy et al. 2013). Since then adversarial examples have been proven to exist in DNN models for various applications such as object detection (Xie et al. 2017), intrusion detection (Yang et al. 2018), and voice recognition (Yuan et al. 2018).

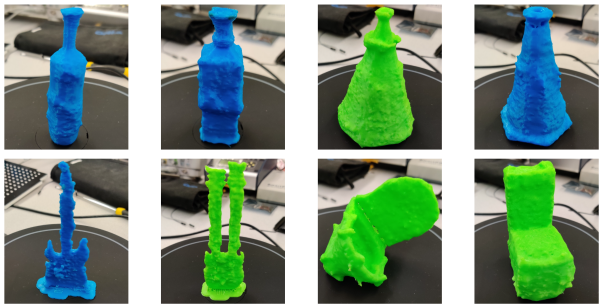

Figure 1: Visualization of the 3D Printed Physical Adversarial Objects Generated by Our Attack Algorithm.

Despite the fact that deep learning models trained to analyze point cloud data are already deployed in safety-crucial applications such as autonomous driving, the threat of potential adversarial attacks on these models is rarely discussed. Though there are some recently proposed adversarial attack algorithms (Xiang, Qi, and Li 2019; Liu, Yu, and Su 2019) targeting the PointNets by adding or altering the points, they fail to pass some trivial defense mechanisms (Zhou et al. 2018). These attack schemes also only concentrate on point clouds; they fail to construct corresponding meaningful objects, not to mention manufacturing these objects physically.

In this work, we focus on adversarial attacks in white-box scenario against PointNet++, a point cloud classification network. To address the limitations of the previous attack algorithms, we propose a novel attack method to generate robust adversarial 3D objects that preserve their properties after physically manufactured by 3D printers. Considering the existing defensive methods designed specifically to defend against 3D adversarial point clouds, we further show

*Corresponding Author.

that our attack can bypass these defense mechanisms with high success rates.

To summarize, we make the following contributions in this paper:

- We propose a novel attack approach to generate 3D adversarial point clouds that can form objects with reasonable shapes in different adversarial settings.
- In our experiments, we illustrate that our attack can bypass existing defenses against adversarial 3D point clouds with high success rates.
- We demonstrate that our attack generates not only robust adversarial points in the digital domain, but also creates printable adversarial objects in physical world.

The source code of this paper is released at https://github.com/jinyier/ai_pointnet_attack.

The remainder of this paper is organized as follows: Section 2 describes related work. Our proposed method is introduced in Section 3. Experiments are given in Section 4, and Section 5 concludes this paper.

2 Related Work

Deep Learning on 3D Points. Point cloud data, as a popular data format that contains 3D coordinates information of points sampled from the surface of physical or virtual objects, are widely applied in 3D vision areas such as industrial modeling, surveying, and autonomous driving. Unlike images with ordered pixels, point cloud data are unordered, making analysis difficult via popular deep learning techniques. The first practical deep learning model to process point cloud data circumvents this issue by using a voxel-based method (Maturana and Scherer 2015; Qi et al. 2016). Since then, more methods based on deep neural networks arise to decrease the computation cost and improve the ability to deal with sparsity. However, these methods can not directly process raw point cloud data, but rely on transformation techniques to convert the raw point cloud data into the form that can be easily processed. Though easy to operate, the performance of these schemes is limited by information loss.

To address those issues, PointNet (Qi et al. 2017a) and its subsequent work, PointNet++ (Qi et al. 2017b), apply max-pooling and transformations to reduce the unordered and dimensionally flexible input data to fixed-length global feature vectors, and by doing so, enable end-to-end neural network learning on raw point cloud data. They demonstrate the robustness of the proposed PointNet by introducing the concept of critical points and upper bounds. The point sets laying between critical points and upper bounds yield the same global features. Thus, PointNet is robust against missing points and random perturbations. Due to their end-to-end learning architecture and high performance, PointNets are widely adopted in different applications such as 3D object detection (Qi et al. 2018; Shi, Wang, and Li 2019) as backbone feature generation networks.

Adversarial Attacks and Defenses in Deep Learning. The phenomenon of adversarial examples was first found by Szegedy (Szegedy et al. 2013), who observed that adding slight but intentionally generated perturbations to legal inputs can mislead deep learning models into making incorrect decisions in 2D image classification. Since then, more algorithms (Goodfellow, Shlens, and Szegedy 2014; Papernot et al. 2016a; Carlini and Wagner 2017; Moosavi-Dezfooli, Fawzi, and Frossard 2016) have been proposed to launch increasingly efficient and effective adversarial attacks in different domains.

As a result of this issue, several approaches have been proposed to defend against adversarial examples. Adversarial training (Tramèr et al. 2017) attempts to create a more robust model during the initial training stage by augmenting the original training dataset with pre-crafted adversarial examples. Defensive distillation (Papernot et al. 2016b) re-trains the model, smoothing the potential adversarial gradients which may be used to craft adversarial inputs. The intuition of distillation is to extract the knowledge of original training data and force the output vectors of the DNN model converge at a large number. As the result adversary will find it hard to distract the output of the DNN model from the correct one. Guo (Guo et al. 2017) explores some preprocessing methods to defend against potentially adversarial inputs, such as compression and transformations. However, these defense mechanisms have each been proven to be ineffective against certain methods of adversarial attacks (Carlini and Wagner 2017).

Adversarial 3D Points. Despite the dangers of adversarial examples being successfully demonstrated in many applications such as: 2D images (Szegedy et al. 2013), automatic speech recognition (Yuan et al. 2018), natural language (Jia and Liang 2017), and network flows (Yang et al. 2018), the vulnerability of deep learning models applied to 3D point data has remained comparatively unexplored. However, recently, Xiang (Xiang, Qi, and Li 2019) proposed the first practical method to generate 3D adversarial point clouds. Since then, Liu (Liu, Yu, and Su 2019) expands this area by proposing further attack methods and metrics. Cao (Cao et al. 2019) studies adversarial 3D point clouds capturable by LiDAR, which is commonly used in autonomous driving.

Though adversarial 3D point clouds for deep learning models have been proven to exist, current techniques to generate them are still at a preliminary stage and thus attacks can be easily detected, removed, or invalidated by simple defense mechanisms such as random sampling and outlier removal, as described by the corresponding defense mechanisms in (Zhou et al. 2018). In addition, current works on generating adversarial 3D data focus on altering original point clouds but neglect the feasibility of reliably manufacturing these adversarial examples in real world. To the best of our knowledge, we are the first to launch robust adversarial attack for 3D machine learning models that can bypass existing defense mechanisms and be used to construct real adversarial objects.

3 Proposed Methodology

3.1 Point-wise Adversarial Perturbation

Our algorithm follows the adversarial attack proposed by Carlini and Wagner (Carlini and Wagner 2017). We apply

their C&W attack with modifications, such as different constraints, for point cloud data and PointNet++ model. Recall that $x \in \mathbb{R}^{n\times 3}$ is the point cloud, and let $\delta \in \mathbb{R}^{n\times 3}$ be the perturbation vectors, we formulate the adversarial attack as an unconstrained optimization problem defined as:

$$\begin{aligned} &\underset{\delta}{\text{argmin}} \;\; f(x+\delta) + c \cdot \|\delta\|_p \\ &\text{with} \;\; f(x) = \max\{\max_{i\neq y'}\{Z(x)_i\} - Z(x)_{y'}, \kappa\} \end{aligned} \tag{1}$$

where y' denotes the attack target, $Z(\cdot)$ is the output of logits layer, and κ is the parameter that controls the attack confidence. The hyper-parameter denoted as c is used to balance the terms in the objective function. In practice, c can be found effectively using binary search. The perturbation δ can be understood as the vectors that describe the *direction* and *magnitude* of shifted points.

Since our ultimate goal is to construct an adversarial object from the original point cloud, we do not limit the number of points that can be altered. All points may be altered as long as the constraints are satisfied and the adversarial point cloud can form a reasonable and printable watertight mesh.

Our objective function can be formally defined as:

$$\underset{\delta}{\text{argmin}} \;\; f(x') + \alpha \cdot D_C(x', x) + \beta \cdot D_K(x') \tag{2}$$

where $x' = x+\delta$, and both α and β are user defined parameters to balance the different constraints. $D_C(\cdot,\cdot)$ and $D_K(\cdot)$, which represent *Chamfer distance* and *kNN distance*, will be described in detail in the following sections.

3.2 Point Cloud Distance Metrics

In previous works, L_2 or L_∞ norm are usually applied to calculate the distances (dis-similarity) between perturbed examples and the original ones. However, considering point cloud data, the L_2 or L_∞ norm may not be good choices of distance metric. Different from images which are fixed dimension grids, point clouds are represented as unordered and unstructured sets, making L_p norm less suitable. Instead, we use Chamfer distance (Fan, Su, and Guibas 2017) as an alternative to measure the distance of two point clouds, from x' to x, which is defined as:

$$D_C(x', x) = \frac{1}{\|x'\|} \sum_{p'\in x'} \left(\min_{p\in x} \|p' - p\|_2^2 \right) \tag{3}$$

In short, it finds for each point p' in x' the closest point p in x (measured by L_2 norm) and averages all the distances.

3.3 Point Cloud Smoothing

Due to the surface reconstruction, the points shifted to relatively far positions will become outliers, making them hard to participate in surface reconstruction and are easily to be removed by defense mechanisms. We propose two different methods to deal with this problem:

K-Nearest Neighbor (kNN) Distance. We calculate the kNN distance in order to limit the distances between adjacent points. First, let $x = \{p_1, p_2, ..., p_n\}$ be the perturbed point cloud. For any point $p \in x$, we define its k-nearest neighbors as the k points in x that have the smallest Euclidean distance to p. Then, we optimize the following objective function:

$$\begin{aligned} D_K(x) &= \frac{1}{\|x\|} \sum_{p\in x} w_p \cdot d_p \\ \text{with} \;\; d_p &= \frac{1}{k} \left(\sum_{p'\in \text{kNN}(p,x)} \|p - p'\|_2^2 \right) \end{aligned} \tag{4}$$

where $w_p = 1$ if d_p is larger than a user defined threshold, or otherwise $w_p = 0$. That is, we minimize d_p to make sure that every point $p \in x$ will be located near its neighbors. This approach is similar to the total variation that is applied on 2D images. The threshold is defined as a multiple of the *standard deviation* σ of all d_p. This leads to smoother point clouds that outperform the non-smoothed ones when constructing objects.

Perturbation Clipping and Projection. Considering real-world conditions, only the point clouds sampled from the "outer" surfaces of the objects are considered. For instance, points sampled from seats or steering wheels inside a car model are discarded, as they are located "inside" the object. The main reason for this is for realism, sensors like LiDAR or 3D scanners cannot obtain the points that are obstructed by other obstacles.

To prevent points shifting into the interior of the object and failing to participate in surface reconstruction, we clip the perturbation vectors that shift the points inside the object by utilizing the inner product of the perturbation vectors and the corresponding normal vectors during the optimization. Let δ_i be the perturbation vector of a point, and N_i be the corresponding normal vector. If $\langle N_i, \delta_i \rangle < 0$, we either set $\delta_i = [0, 0, 0]$ (*i.e.,* reset the perturbation) or project δ_i onto $(N_i \times \delta_i) \times N_i$ (*i.e.,* only keep the magnitude that is vertical to N_i), where $\langle \cdot, \cdot \rangle$ and $\times$ are inner product and cross product of vectors, respectively. This rule cannot perfectly handle all situations since the shape of an object is not always regular. However, it does help limit the perturbation vectors to our preferred directions in general cases.

We also utilize the infinity norm L_∞ to limit the magnitude of the perturbations. If the magnitude $\|\delta_i\|_2$ is greater than a given threshold ℓ_∞, it will be scaled in order to satisfy the constraint by:

$$\delta_i^{new} = \frac{\delta_i}{\|\delta_i\|_2} \cdot \ell_\infty \tag{5}$$

3.4 Random Sampling

Due to the unordered property of point cloud data, the PointNets are designed and trained with specific network architectures and data augmentations to make sure that the models will not be easily misled by just using another subset of points from the same object. Therefore, the PointNets are expected to be robust against randomized input data, increasing their resilience against adversarial attacks. To deal with this property, we randomly pick a subset of points in each optimization step as the input to the model to simulate such

Table 1: Accuracy (%) of Our Trained PointNet++ Models Evaluated Using the ModelNet40 Dataset.

Models	Training	Testing	All Examples
SSG-1024	98.10	89.59	96.39
SSG-2048	97.78	88.49	95.91

procedure. This also increases the robustness of the point sets against cases where only a portion of the points are used as inputs.

3.5 Surface Reconstruction

After the attack, the perturbed point cloud data are further processed to construct meaningful objects. In this work, we choose Screened Poisson Surface Reconstruction (Kazhdan and Hoppe 2013) as the surface reconstruction algorithm. Given a 3D point cloud x and its corresponding normal vectors N, the algorithm will output a watertight mesh described by a set of vertices and faces. Since most of the points are shifted from their original locations during attack, we estimate their normal vectors by a simple approach. Let N_p be the normal vector of the point p in original point cloud x, we calculate the estimated one by:

$$N'_{p'} = \frac{1}{k} \sum_{p \in \mathrm{kNN}(p', x)} N_p \tag{6}$$

where p' is a point in perturbed point cloud x', and $N'_{p'}$ is its corresponding new normal vector.

Finally, given the reconstructed meshes, the adversarial point clouds are sampled and classified using PointNet++ to evaluate our attack performance.

4 Experimental Results

We generate adversarial examples from various 3D models using our proposed algorithm and evaluate the attack results in different scenarios with PointNet++ as the victim network. Additionally, existing defense mechanisms are also tested against our examples. Several adversarial objects are 3D printed, scanned by 3D scanners, and then the resulting point clouds are classified by PointNet++ to demonstrate that our attack remains effective even in physical world.

4.1 Experimental Setup

Datasets. We use the ModelNet40 dataset (Wu et al. 2015) for our experiments, including training, testing the victim models, and generating adversarial examples. This dataset contains 12,311 CAD models with 40 common object categories in real world. We use the official splits, where 9,843 examples are used for training, and the remaining 2,468 examples are used for testing. For adversarial attacks, we randomly choose a subset of data in both training and testing set as the "clean" point clouds with certain conditions, which will be described in detail later.

Victim Models. We choose PointNet++ as our target model. The models are trained using the dataset described previously with network architectures and hyper-parameters proposed by the authors of PointNet++. In our experiments, two classification models are trained in different settings without considering normal vectors to classify 40 categories:

- SSG-1024: SSG with 1024 input points
- SSG-2048: SSG with 2048 input points

where SSG is a grouping strategy introduced in PointNet++ called *single scale grouping*. The details of our trained models are described in Section 4.2.

Adversarial Settings. In this work, both targeted and untargeted attacks are considered. We pick a subset of classes from ModelNet40 as target labels, which are *airplane*, *bottle*, *car*, *chair*, *monitor*, and *sofa*. For each class, we randomly choose 20 benign examples that can be correctly classified by PointNet++, and the surface can be correctly reconstructed using the original point clouds and normal vectors.

For untargeted attack, we set the attack goal to misclassify the point clouds to any class other than the groundtruth (*i.e.,* there are 39 possible targets). For targeted attack, we consider two attack targets: *most likely* and *random target*. The former is the class predicted with probability only less than the groundtruth's, and the later is a randomly chosen class from the 38 remaining classes (*i.e.,* excluding the groundtruth and most likely classes). We generate 360 adversarial examples for each victim model. For all experiments, the points used for classification will be a randomly selected subset of the point clouds (*e.g.,* randomly pick 1024 from 10,000 points, which are sampled from objects). Due to this randomized procedure, the points will be re-chosen for several times (16 times in our experiments) for classification, and all the results will be averaged.

Implementation Details. In our implementation, we only consider the Chamfer distance from the *adversarial* point cloud to the *original* one. The distance from the *original* one to the *adversarial* one is not considered. Similarly, only the points selected as the input of the model (*i.e.,* either 1024 or 2048 points in our settings) are used for kNN distance calculation. The threshold of kNN distance is set to $1.1 \times \sigma$, which is described in (4), and only the distances greater than the threshold are penalized during the optimization process. Both Chamfer distance and kNN distance are calculated using the point clouds which are not normalized.

Environment and Equipment. The attack algorithm is carried on a server with Intel 9900K CPU, two NVIDIA RTX 2080 Ti graphic cards and 64GB RAM. The code is written in Python programming language with TensorFlow framework. The physical adversarial objects are 3D printed by FLASHFORGE CreaterPro 3D printer and re-scanned as meshes by EinScan-SE 3D scanner.

4.2 Evaluation of the Trained Models

For each model, we evaluate it using training set (which contains 9,843 point sets), testing set (which contains 2,468 point sets), and the whole dataset. The results can be found in Table 1. The models we trained both obtain similar performance with the ones proposed in the PointNet++ paper.

Table 2: Attack Success Rates (%) against Different PointNet++ Models with (a) Untargeted Attack (b) Most Likely Attack (c) Random Target Attack.

	SSG-1024						SSG-2048					
Attack	airplane	bottle	car	chair	monitor	sofa	airplane	bottle	car	chair	monitor	sofa
(a)	100.0	100.0	100.0	99.69	100.0	100.0	100.0	99.38	100.0	98.75	100.0	100.0
(b)	100.0	97.50	97.50	93.13	95.00	100.0	99.38	97.50	84.69	94.06	98.75	100.0
(c)	95.31	16.25	86.25	72.19	78.75	81.56	90.62	13.44	82.81	87.19	64.69	59.06

Table 3: Attack Success Rates (%) against Different PointNet++ Models Using Closest Sampling Strategy with (a) Untargeted Attack (b) Most Likely Attack (c) Random Target Attack.

	SSG-1024						SSG-2048					
Attack	airplane	bottle	car	chair	monitor	sofa	airplane	bottle	car	chair	monitor	sofa
(a)	100.0	100.0	95.31	97.50	79.69	99.38	99.06	99.06	88.44	88.44	37.50	79.37
(b)	100.0	95.00	83.75	87.81	86.25	100.0	96.88	90.00	54.37	80.94	48.75	86.25
(c)	94.69	14.06	73.75	55.63	66.25	60.31	72.19	7.81	49.06	58.13	23.44	21.25

4.3 Adversarial Attack Evaluation

We first evaluate the adversarial examples by directly classifying them using our trained models. The hyper-parameters we used for attack are: $\alpha = 5$, $\beta = 3$, $\ell_\infty = 0.1$, $\kappa = -15$, which are defined in Equations (1), (2), and (5). The hyper-parameters are chosen to balance the kNN and Chamfer distance. The misclassification loss (*i.e.,* the $f(\cdot)$ in (2)) will be ignored if it reaches the lower bound of the attack confidence, denoted as κ. Thus, it would not dominate the whole optimization process. For targeted attack, the target is chosen either by first classifying the original point cloud, and then choosing the class with second highest probability (for *most likely*), or randomly (for *random target*). Once the target is determined, it will not be changed during the attack process.

The adversarial examples are generated using the trained models, and the corresponding success rates are shown in Table 2. We can see that the success rates vary greatly between different original-target pairs, such as the random target attack with class "bottle". One possible reason is that in our adversarial settings, the perturbation budget is not large enough to make such examples be misclassified. We believe that the success rates will increase with more flexible constraints. However, if the points are allowed to shift in large distances, humans will be more likely to perceive differences between them and the original objects. Another factor that may affect the attack difficulty is the grouping and pooling methods introduced in PointNet++ architecture. These mechanisms could make the models more robust by training on features in different scales, so that the model will become less sensitive to small perturbations.

4.4 Surface Reconstruction and Re-sampling

The perturbed point clouds are further processed to reconstruct object surfaces. We apply the Screened Poisson Surface Reconstruction algorithm (Kazhdan and Hoppe 2013) using the implementation provided by MeshLab. After the reconstruction, isolated components will be removed if the number of faces is less than a given threshold (1024 faces in our experiments) to make the object more smooth. This ensures that the constructed meshes will not contain many awkwardly connected components, which would cause them to stand out in real world. Lastly, the points are sampled from the meshes according to two different strategies:

- Closest: Choose the points on the surface that have the shortest distances to the perturbed point clouds.
- Random: Choose points from the mesh surface randomly.

For each mesh, we sample 10,000 points using above mechanisms, and a subset of points will be randomly chosen as the input to the classification models. The attack success rates are shown in Table 3 and Table 4 with closest and random sampling, respectively. Visualization of the adversarial point clouds with closest sampling can be found in Figure 2.

The attack remains effective in the closest sampled point clouds. Although some adversarial properties are lost after the reconstruction, the majority of the attacks achieves reasonable success rates. On the other hand, the success rates occasionally drop in the randomly sampled cases. This indicates that the adversarial attack may rely on the adversarial points in particular regions or in non-uniform distributions, which can be considered more vulnerable. Thus, if the points are forced to be distributed uniformly along the object surface, the attack will become less effective after the grouping and pooling processes, since most of the perturbed points have been shifted away from the vulnerable regions.

4.5 Existing Defense Mechanisms

Considering several existing defense approaches against 3D adversarial attacks, we further evaluate our adversarial examples using the following mechanisms:

- Apply random rotations on point clouds, which is implemented in the official PointNet++ codebase.
- Apply kNN outlier removal introduced in (Zhou et al. 2018). It is denoted as *Statistical Outlier Removal* (SOR) in the proposed paper.

Table 4: Attack Success Rates (%) against Different PointNet++ Models Using Random Sampling Strategy with (a) Untargeted Attack (b) Most Likely Attack (c) Random Target Attack.

	SSG-1024						SSG-2048					
Attack	airplane	bottle	car	chair	monitor	sofa	airplane	bottle	car	chair	monitor	sofa
(a)	89.38	92.50	18.12	58.75	15.62	26.87	42.19	67.19	10.94	41.25	2.81	20.31
(b)	73.12	94.37	14.37	47.81	21.25	30.00	40.31	42.81	0.63	38.44	3.75	19.69
(c)	27.81	0.0	2.81	2.50	7.19	0.31	5.94	0.0	0.0	5.31	0.0	0.0

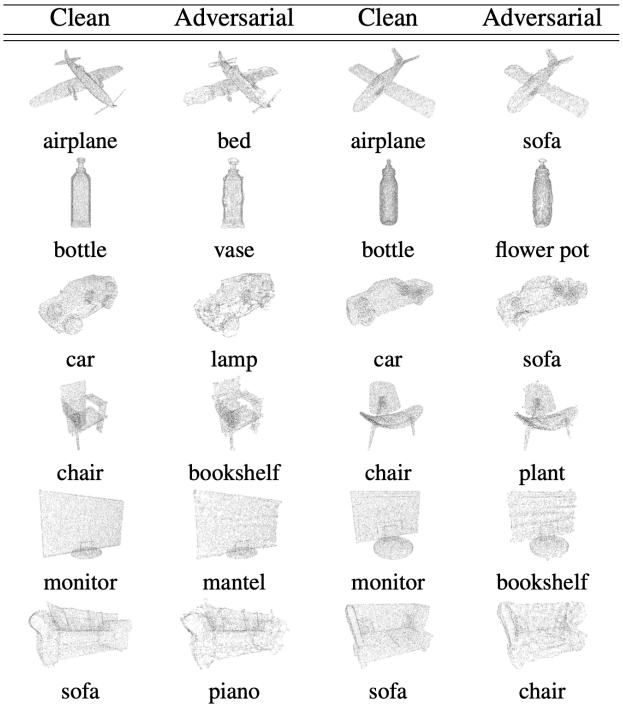

Figure 2: Visualization of the Adversarial Point Clouds Generated by Our Attack Algorithm.

- Statistical defense through applying random Gaussian noise, which is introduced in (Yang et al. 2019).

Note that randomized input (*i.e.,* randomly choose a subset of points which are sampled from the meshes as input to the models) is considered in all our experiments. The defensive mechanisms are evaluated using SSG-1024 model with closest sampling due to the higher success rates, and we believe that it is a better choice to evaluate the defense mechanisms against our adversarial examples.

Voting Classification by Rotations. This mechanism is implemented in official PointNet++ codebase, which applies rotations on point clouds before they are fed into the models. Although PointNets are designed to be invariant to point ordering, they are sensitive to rotations. We evaluate our examples to see whether our point clouds are robust to such data augmentations. In our experiments, numbers of rotations from 2 to 10 are tested, and the results are shown in Table 5. Since this procedure is not considered during our attack process, the success rates drop by about 10–30%, especially for the models with lower success rates originally. However, we still achieve reasonable and stable success rates in most test cases.

Table 5: Attack Success Rates (%) Using Voting Classification with SSG-1024.

#Rotations	1	2	3	4	5
Untargeted	95.31	90.10	87.86	86.88	86.88
Most Likely	92.14	81.93	79.58	78.12	77.71
Random	60.78	39.53	32.92	32.29	30.36
#Rotations	6	7	8	9	10
Untargeted	86.46	85.21	86.35	86.09	86.51
Most Likely	76.04	76.93	76.56	76.15	75.73
Random	30.83	29.74	31.35	29.11	29.84

K-Nearest Neighbor Outlier Removal. This defense aims to weaken the adversarial attacks by removing the points that locate relatively far away from other points. It is considered that prior adversarial attacks mostly rely on those "outlier" points, so that this removal process should invalidate the attacks. As a quick review, for each point p, its kNNs can be found by utilizing Euclidean distances, and let the averaged distance be d_p. Then, the mean μ and standard deviation σ of all d_p are calculated. A point p will be removed if:

$$d_p > \mu + \alpha \times \sigma \tag{7}$$

To find the hyper-parameters in (7), we perform grid search on both k and α. For k, we test the values $2, 4, 6, 8, 10$, and for α, we test the values $0.0, 0.2, 0.4, ..., 1.0$. The results are shown in Table 6 with *untargeted*, *most likely*, and *random target* attacks. We believe that, in addition to kNN smoothing, applying randomized input during attacks (*i.e.,* use different subsets of points in each iteration) also helps us bypass this defense mechanism. Note that k greater than 10 are not evaluated because the classification accuracy using benign examples will be dropped lower than 80%, according to (Zhou et al. 2018) (*e.g.,* for $k = 10$ and $\alpha = 0.5$, the accuracy of benign examples is lower than 0.8).

Statistical Defense Through Random Gaussian Noise. In contrast to the previous two defenses, which try to make the adversarial examples be correctly classified, this method aims to "detect" whether an input is adversarial or not. In our case, this mechanism can be considered as a "black-box" defense as we do not take this into consideration during our attack. As a quick review, let x be an input point cloud (which can be benign or adversarial), we add random Gaus-

Table 6: Attack Success Rates (%) against kNN Outlier Removal Defense with SSG-1024.

Untargeted Attack

k \ α	0.0	0.2	0.4	0.6	0.8	1.0
2	99.79	99.74	99.74	99.79	99.32	98.96
4	99.53	99.90	99.90	99.38	99.69	99.58
6	99.22	99.58	99.74	99.58	99.79	99.69
8	98.85	99.27	99.43	99.64	99.74	99.79
10	98.39	98.59	98.80	99.32	99.32	99.53

Most Likely Attack

k \ α	0.0	0.2	0.4	0.6	0.8	1.0
2	94.84	95.57	96.04	95.68	95.83	95.52
4	92.60	94.69	95.57	95.89	95.47	95.36
6	90.31	92.08	93.02	94.48	94.90	95.05
8	85.36	89.79	91.72	92.66	94.11	95.21
10	82.45	86.82	89.90	91.56	92.40	93.65

Random Target Attack

k \ α	0.0	0.2	0.4	0.6	0.8	1.0
2	65.26	67.66	67.40	67.08	68.13	67.40
4	61.88	64.74	66.56	67.55	67.40	68.54
6	56.30	60.62	63.75	66.87	67.50	65.36
8	50.31	56.35	61.67	63.65	65.10	66.09
10	43.07	50.94	58.13	60.52	62.97	64.58

sian noise to generate a set of perturbed point clouds:

$$x'_i = x + \rho_i \quad \text{s.t.} \quad \rho_i \sim N(0, \sigma^2) \tag{8}$$

where the noise ρ_i is i.i.d. sampled from the Gaussian distribution $N(0, \sigma^2)$. It is considered that adding non-directional noise helps the inputs escape from the narrow adversarial subspace, leading to unstable attack results. Then, given a set of perturbed point clouds x', we let o' be the output set of the classification model, and o'_{ij} be the confidence of class j of point cloud x'_i. Lastly, the *Set-Indiv Variance Measurement* $SIV(\cdot)$ is defined as:

$$SIV(o') = \frac{1}{N_c} \sum_{k=1}^{N_c} \mathrm{Var}_{i \in 1,2,\ldots,m}(o'_{ik}) \tag{9}$$

where N_c is the number of classes (40 in our case), and m denotes the times we add random noise (10 in our case). In our experiments, we evaluate this detection method in different *Defense Detection Rate (DDR)* ($t\%$) settings as defined in (Yang et al. 2019), which measures how many adversarial examples are detected while $t\%$ of benign examples are incorrectly rejected. The false negative rates (FNR) (*i.e.,* $1.0 - \text{DDR}$, the success rate of adversarial examples that can bypass the defense) are reported in Table 7 using *untargeted*, *most likely*, and *random target* attacks in two different DDR settings: $t = 5$ and $t = 10$. Only the *correctly* classified

Table 7: False Negative Rates (%) of Statistical Defense Using Gaussian Noise with SSG-1024 and Successful Adversarial Examples.

Untargeted Attack

σ^2	1e−5	5e−5	1e−4	5e−4	1e−3
FNR (5%)	87.07	93.91	86.96	88.50	85.34
FNR (10%)	76.72	84.35	76.52	71.68	76.72
σ^2	5e−3	1e−2	5e−2	1e−1	5e−1
FNR (5%)	83.48	75.22	98.25	96.58	92.79
FNR (10%)	71.30	63.72	95.61	86.32	90.09

Most Likely Attack

σ^2	1e−5	5e−5	1e−4	5e−4	1e−3
FNR (5%)	88.60	89.29	92.04	86.84	90.57
FNR (10%)	82.46	85.71	86.73	83.33	86.79
σ^2	5e−3	1e−2	5e−2	1e−1	5e−1
FNR (5%)	87.39	77.06	93.81	95.37	98.20
FNR (10%)	83.78	70.64	89.38	87.04	93.69

Random Target Attack

σ^2	1e−5	5e−5	1e−4	5e−4	1e−3
FNR (5%)	79.12	63.64	66.67	62.92	53.33
FNR (10%)	53.85	52.27	51.61	51.69	43.33
σ^2	5e−3	1e−2	5e−2	1e−1	5e−1
FNR (5%)	53.49	32.22	96.70	97.59	96.43
FNR (10%)	43.02	21.11	91.21	96.39	88.10

benign examples in ModelNet40 dataset (11,621 examples while using SSG-1024 model) are used to define the threshold, and only the examples that are misclassified (*i.e.,* the attack succeeded) are considered as "adversarial".

We can see that in different adversarial attacks, the detection rates vary greatly. For the worst case, about 30–40% of the adversarial examples are rejected in *untargeted* and *most likely* attack, while about 80% of them are rejected in *random targeted* attack. One possible explanation is that due to our adversarial constraints, there exist some attack targets that are relatively hard to succeed, as shown in previous experiments such as Table 2 or Table 3, which lead to easily detectable attack results when subjected to random non-directional noise. This phenomenon is demonstrated by our attacks with lower success rates, which indicate unstable adversarial examples, suggesting high susceptibility to Gaussian noise, leading to high detection rates. If this is the case, attacking with more flexible constraints (*e.g.,* more budget for adversarial perturbations) may increase the robustness of the adversarial examples against this defense. However, the attacks will become more perceptible and can be easily distinguished by humans.

4.6 Physical Adversarial 3D Objects

We physically print 10 perturbed adversarial objects using 3D printers, and then re-scan them with 3D scanners. For each object, we randomly sample 50,000 points from the re-scanned mesh for further classification. Visualization of the

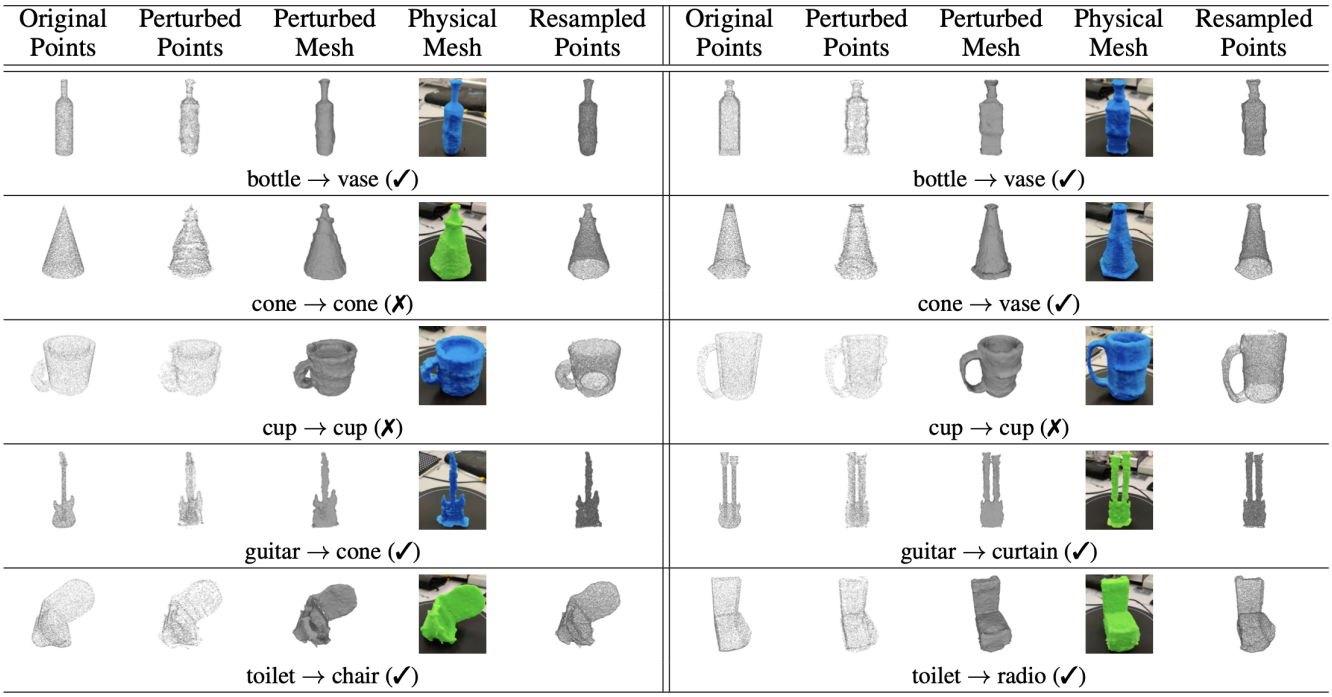

Figure 3: Visualization of the Physical 3D Objects and Corresponding Point Clouds with Classification Results.

Table 8: Attack Success Rates against Existing Defenses Using Point Sets Sampled from Physical Objects.

Voting	Outlier Removal	Statistical[1]
86.71%	80.47%	71.43%, 42.86%

[1]Success rates in both $t = 5$ and $t = 10$ settings.

Table 9: Comparison of Existing 3D Adversarial Attacks.

Approach	Success Rate	Defense	Physical
Xiang, et al.	100%		
Liu, et al.	91.2–96.6%	✓	
Ours	60.8–95.3%	✓	✓

adversarial objects and point clouds with classification results can be found in Figure 3. We can find that the attacks mostly fail for complex objects, where most of the points are lost during the re-scanning process. This also happens when the object has "inner" surfaces such as in "bowl" or "cup" objects, because the 3D scanner cannot perfectly capture the points that are located on the interior of an object, resulting in a loss of adversarial properties. Another challenge is the random sampling procedure, as shown in Table 4. The re-sampling procedure has a large impact on the attack performance as it forces the points to be distributed more uniformly along the object, and many points located in the vulnerable regions of the point cloud are lost.

We use the successful results (7 objects) to evaluate the defense mechanisms previously mentioned using the hyper-parameters we found. For voting classification, we apply 10 rotations. For outlier removal, we set $k = 10$ and $\alpha = 0.0$. For statistical defense using Gaussian noise, we let $\sigma^2 = 0.01$. The results can be found in Table 8, which shows that our attack can still achieve reasonable success rates while considering existing defenses.

To summarize, we address the limitations that remain unresolved in previous works, and the major differences between our work and previous studies are shown in Table 9. Although it is difficult to compare the success rates directly due to different adversarial settings, the experiments shown that our attack can still obtain comparable success rates while considering existing defensive mechanisms and physical constraints in real world.

5 Conclusion

In this paper, we presented an attack algorithm that generates adversarial 3D objects against PointNet++, a widely-used 3D object classification model. Different metrics were proposed to deal with the unsolved constraints in previous work, including random sampling and surface reconstruction. In the experiments, our algorithm was tested under several conditions and defensive mechanisms which are designed especially for adversarial 3D examples, and the results showed that our attack can bypass such mechanisms with high success rates. In addition, several physical objects were 3D printed and evaluated. The results showed that our attack can successfully generate physical objects while preserving adversarial properties.

References

Cao, Y.; Xiao, C.; Cyr, B.; Zhou, Y.; Park, W.; Rampazzi, S.; Chen, Q. A.; Fu, K.; and Mao, Z. M. 2019. Adversarial sensor attack on lidar-based perception in autonomous driving. *arXiv preprint arXiv:1907.06826*.

Carlini, N., and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *Proceedings of the Security and Privacy (S&P) on 2017 IEEE Symposium*. IEEE.

Fan, H.; Su, H.; and Guibas, L. J. 2017. 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*, 605–613.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Guo, C.; Rana, M.; Cisse, M.; and van der Maaten, L. 2017. Countering adversarial images using input transformations. *arXiv preprint arXiv:1711.00117*.

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

Kazhdan, M., and Hoppe, H. 2013. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)* 32(3):29.

Liu, D.; Yu, R.; and Su, H. 2019. Extending adversarial attacks and defenses to deep 3d point cloud classifiers. *arXiv preprint arXiv:1901.03006*.

Maturana, D., and Scherer, S. 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 922–928. IEEE.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.

Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swami, A. 2016a. The limitations of deep learning in adversarial settings. In *Proceedings of the Security and Privacy (S&P) on 2016 IEEE European Symposium*. IEEE.

Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; and Swami, A. 2016b. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (S&P), 2016 IEEE Symposium on*, 582–597. IEEE.

Qi, C. R.; Su, H.; Nießner, M.; Dai, A.; Yan, M.; and Guibas, L. J. 2016. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5648–5656.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 652–660.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, 5099–5108.

Qi, C. R.; Liu, W.; Wu, C.; Su, H.; and Guibas, L. J. 2018. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 918–927.

Shi, S.; Wang, X.; and Li, H. 2019. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–779.

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

Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2017. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*.

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

Xiang, C.; Qi, C. R.; and Li, B. 2019. Generating 3d adversarial point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9136–9144.

Xie, C.; Wang, J.; Zhang, Z.; Zhou, Y.; Xie, L.; and Yuille, A. 2017. Adversarial examples for semantic segmentation and object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 1369–1378.

Yang, K.; Liu, J.; Zhang, C.; and Fang, Y. 2018. Adversarial examples against the deep learning based network intrusion detection systems. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, 559–564. IEEE.

Yang, J.; Zhang, Q.; Fang, R.; Ni, B.; Liu, J.; and Tian, Q. 2019. Adversarial attack and defense on point sets. *arXiv preprint arXiv:1902.10899*.

Yuan, X.; Chen, Y.; Zhao, Y.; Long, Y.; Liu, X.; Chen, K.; Zhang, S.; Huang, H.; Wang, X.; and Gunter, C. A. 2018. Commandersong: A systematic approach for practical adversarial voice recognition. In *27th USENIX Security Symposium (USENIX Security 18)*, 49–64.

Zhou, H.; Chen, K.; Zhang, W.; Fang, H.; Zhou, W.; and Yu, N. 2018. Deflecting 3d adversarial point clouds through outlier-guided removal. *arXiv preprint arXiv:1812.11017*.