AdvPC: Transferable Adversarial Perturbations on 3D Point Clouds

Abdullah Hamdi, Sara Rojas, Ali Thabet, and Bernard Ghanem

King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia
{abdullah.hamdi, sara.rojasmartinez, ali.thabet,
bernard.ghanem}@kaust.edu.sa

Abstract. Deep neural networks are vulnerable to adversarial attacks, in which imperceptible perturbations to their input lead to erroneous network predictions. This phenomenon has been extensively studied in the image domain, and has only recently been extended to 3D point clouds. In this work, we present novel data-driven adversarial attacks against 3D point cloud networks. We aim to address the following problems in current 3D point cloud adversarial attacks: they do not transfer well between different networks, and they are easy to defend against via simple statistical methods. To this extent, we develop a new point cloud attack (dubbed AdvPC) that exploits the input data distribution by adding an adversarial loss, after Auto-Encoder reconstruction, to the objective it optimizes. AdvPC leads to perturbations that are resilient against current defenses, while remaining highly transferable compared to state-of-the-art attacks. We test AdvPC using four popular point cloud networks: PointNet, PointNet++ (MSG and SSG), and DGCNN. Our proposed attack increases the attack success rate by up to 40% for those transferred to unseen networks (transferability), while maintaining a high success rate on the attacked network. AdvPC also increases the ability to break defenses by up to 38% as compared to other baselines on the ModelNet40 dataset. The code is available at https://github.com/ajhamdi/AdvPC.

1 Introduction

Deep learning has shown impressive results in many perception tasks. Despite its performance, several works show that deep learning algorithms can be susceptible to adversarial attacks. These attacks craft small perturbations to the inputs that push the network to produce incorrect outputs. There is significant progress made in 2D image adversarial attacks, where extensive work shows diverse ways to attack 2D neural networks [23,6,11,18,4,2,35,8,7]. In contrast, there is little focus on their 3D counterparts [31,38,37,25]. 3D point clouds captured by 3D sensors like LiDAR are now widely processed using deep networks for safety-critical applications, including but not limited to self-driving [3,27]. However, as we show in this paper, 3D deep networks tend to be vulnerable to input perturbations, a fact that increases the risk of using them in such applications. In this paper, we present a novel approach to attack deep learning algorithms applied to 3D point clouds with a primary focus on attack transferability between networks.

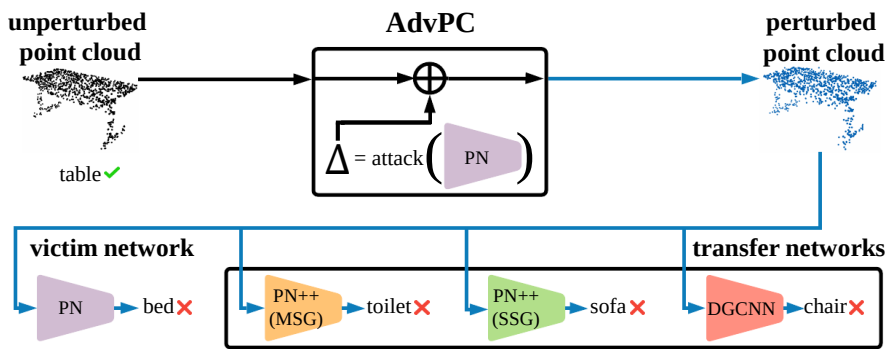

Fig. 1: **Transferable Adversarial Perturbations on 3D point clouds**: Generating adversarial attacks to fool PointNet [21](PN) by perturbing a Table point cloud. The perturbed 3D object not only forces PointNet to predict an incorrect class, but also induces misclassification on other unseen 3D networks (PointNet++ [22], DGCNN [29]) that are not involved in generating the perturbation. Fooling unseen networks poses a threat to 3D deep vision models.

The concept of attack transferability has been extensively studied in the 2D image domain [17,19,20]. Transferability allows an adversary to fool any network, without access to the network's architecture. Clearly, transferable attacks pose a serious security concern, especially in the context of deep learning model deployment. In this work, the goal is to generate adversarial attacks with network-transferability, *i.e.* the attack to a given point cloud is generated using a single and accessible *victim* network, and the perturbed sample is directly applied to an unseen and inaccessible *transfer* network.Accessibility here refers to whether the parameters and architecture of the network are known, while optimizing the attack (white-box). Fig. 1 illustrates the concept of transferability. The perturbation generated by our method for a 3D point cloud not only flips the class label of a victim network to a wrong class (*i.e.* it is adversarial), but it also induces a misclassification for the transfer networks that are not involved in generating the perturbation (*i.e.* it is transferable).

Very few adversarial attacks have been developed for 3D point clouds. The first method was introduced by Xiang *et. al.* [31] and it proposes point perturbation and adversarial point generation as two attack modes. More recently, Tsai *et. al.* [25] proposed to make point cloud attacks more smooth and natural by incorporating a K-Nearest Neighbor (KNN) loss on the points, thus making the attacks physically realizable. We identify two main shortcomings in current 3D adversarial perturbations methods [31,25]. First, their attacks are unsuccessful in the presence of simple defenses, such as Statistical Outlier Removal [38]. Second, they are limited to the victim network and do not transfer well to other networks [31]. In contrast, our work not only focuses on adversarial perturbations that are significantly more resilient against currently available point cloud defenses, but also on those that transfer well between different point cloud networks.

To generate more transferable attacks, we use a point cloud Auto-Encoder (AE), which can effectively reconstruct the unperturbed input after it is perturbed, and then add a data adversarial loss. We optimize the perturbation added to the input to fool the classifier *before* it passes through the AE (regular adversarial loss) and *after* it passes through the AE (data adversarial loss). In doing so, the attack tends to be less dependent on the victim network, and generalizes better to different networks. Our attack is dubbed "AdvPC", and our full pipeline is optimized end-to-end from the classifier output to the perturbation. The AE learns the natural distribution of the data to generalize the attack to a broader range of unseen classifiers [26], thus making the attack more dangerous. Our attacks surpass state-of-the-art attacks [31,25] by a large margin (up to 40%) on point cloud networks operating on the standard ModelNet40 dataset [30] and for the same maximum allowed perturbation norms (norm-budgets).

Contributions. Our contributions are two-fold. **(1)** We propose a new pipeline and loss function to perform transferable adversarial perturbations on 3D point clouds. By introducing a data adversarial loss targeting the victim network after reconstructing the perturbed input with a point cloud AE, our approach can be successful in both attacking the victim network and transferring to unseen networks. Since the AE is trained to leverage the point cloud data distribution, incorporating it into the attack strategy enables better transferability to unseen networks. To the best of our knowledge, we are the first to introduce network-transferable adversarial perturbations for 3D point clouds. **(2)** We perform extensive experiments under constrained norm-budgets to validate the transferability of our attacks. We transfer our attacks between four point cloud networks and show superiority against the state-of-the-art. Furthermore, we demonstrate how our attacks outperform others when targeted by currently available point cloud defenses.

2 Related Work

2.1 Deep Learning for 3D Point Clouds

PointNet [21] paved the way as the first deep learning algorithm to operate directly on 3D point clouds. PointNet computes point features independently, and aggregates them using an order invariant function like max-pooling. An update to this work was PointNet++ [22], where points are aggregated at different 3D scales. Subsequent works focused on how to aggregate more local context [5] or on more complex aggregation strategies like RNNs [9,33]. More recent methods run convolutions across neighbors of points, instead of using point-wise operations [29,15,24,13,12,15,28,14]. Contrary to PointNet and its variants, these works achieve superior recognition results by focusing on local feature representation. In this paper and to evaluate/validate our adversarial attacks, we use three point-wise networks, PointNet [21] and PointNet++ [22] in single-scale (SSG) and multi-scale (MSG) form, and a Dynamic Graph convolutional Network, DGCNN [29]. We study the sensitivity of each network to adversarial perturbations and show the transferability of AdvPC attacks between the networks.

2.2 Adversarial Attacks

Pixel-based Adversarial Attacks. The initial image-based adversarial attack was introduced by Szegedy *et. al.* [23], who cast the attack problem as optimization with pixel perturbations being minimized so as to fool a trained classifier into predicting a wrong class label. Since then, the topic of adversarial attacks has attracted much attention [6,11,18,4,16]. More recent works take a learning-based approach to the attack [19,20,36]. They train a neural network (adversary) to perform the attack and then use the trained adversary model to attack unseen samples. These learning approaches [19,20,36] tend to have better transferability properties than the optimizations approaches [6,11,18,4,16], while the latter tend to achieve higher success rates on the victim networks. As such, our proposed AdvPC attack is a *hybrid* approach, in which we leverage an AE to capture properties of the data distribution but still define the attack as an optimization for each sample. In doing so, AdvPC captures the merits of both learning *and* optimization methods to achieve high success rates on the victim networks as well as better transferability to unseen networks.

Adversarial Attacks in 3D. Several adversarial attacks have moved beyond pixel perturbations to the 3D domain. One line of work focuses on attacking image-based CNNs by changing the 3D parameters of the object in the image, instead of changing the pixels of the image [8,35,2,7,32]. Recently, Xiang *et. al.* [31] developed adversarial perturbations on 3D point clouds, which were successful in attacking PointNet [21]; however, this approach has two main shortcomings. First, it can be easily defended against by simple statistical operations [38]. Second, the attacks are non-transferable and only work on the attacked network [31,38]. In contrast, Zheng *et. al.* [37] proposed dropping points from the point cloud using a saliency map, to fool trained 3D deep networks. As compared to [37], our attacks are modeled as an optimization on the additive perturbation variable with a focus on point perturbations instead of point removal. As compared to [31], our AdvPC attacks are significantly more successful against available defenses and more transferable beyond the victim network, since AdvPC leverages the point cloud data distribution through the AE. Concurrent to our work is the work of Tsai *et. al.* [25], in which the attack is crafted with KNN loss to make smooth and natural shapes. The motivation of their work is to craft natural attacks on 3D point clouds that can be 3D-printed into real objects. In comparison, our novel AdvPC attack utilizes the data distribution of point clouds by utilizing an AE to generalize the attack.

Defending Against 3D Point Cloud Attacks. Zhou *et. al.* [38] proposed a Statistical Outlier Removal (SOR) method as a defense against point cloud attacks. SOR uses KNN to identify and remove point outliers. They also propose DUP-Net, which is a combination of their SOR and a point cloud up-sampling network PU-Net [34]. Zhou *et. al.* also proposed removing unnatural points by Simple Random Sampling (SRS), where each point has the same probability of being randomly removed. Adversarial training on the attacked point cloud is also proposed as a mode of defense by [31]. Our attacks surpass state-of-the-art

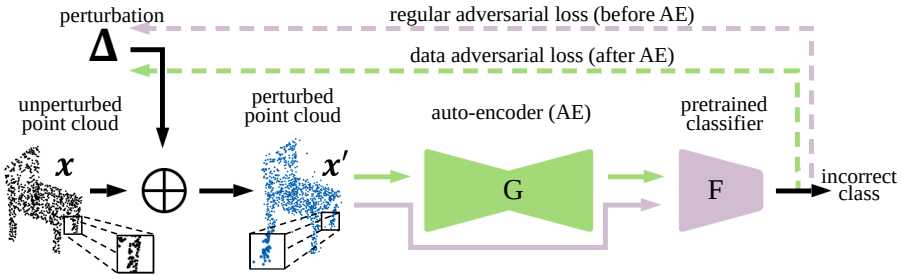

Fig. 2: **AdvPC Attack Pipeline:** We optimize for the constrained perturbation variable $\boldsymbol{\Delta}$ to generate the perturbed sample $\mathcal{X}' = \mathcal{X} + \boldsymbol{\Delta}$. The perturbed sample fools a trained classifier $\mathbf{F}$ (*i.e.* $\mathbf{F}(\mathcal{X}')$ is incorrect), and at the same time, if the perturbed sample is reconstructed by an Auto-Encoder (AE) $\mathbf{G}$, it too fools the classifier (*i.e.* $\mathbf{F}(\mathbf{G}(\mathcal{X}'))$ is incorrect). The AdvPC loss for network $\mathbf{F}$ is defined in Eq (6) and has two parts: network adversarial loss (*purple*) and data adversarial loss (*green*). Dotted lines are gradients flowing to the perturbation variable $\boldsymbol{\Delta}$.

attacks [31,25] on point cloud networks by a large margin (up to 38%) on the standard ModelNet40 dataset [30] against the aforementioned defenses [38].

3 Methodology

The pipeline of AdvPC is illustrated in Fig. 2. It consists of an Auto-Encoder (AE) $\mathbf{G}$, which is trained to reconstruct 3D point clouds and a point cloud classifier $\mathbf{F}$. We seek to find a perturbation variable $\boldsymbol{\Delta}$ added to the input $\mathcal{X}$ to fool $\mathbf{F}$ before *and* after it passes through the AE for reconstruction. The setup makes the attack less dependent on the victim network and more dependent on the data. As such, we expect this strategy to generalize to different networks. Next, we describe the main components of our pipeline: 3D point cloud input, AE, and point cloud classifier. Then, we present our attack setup and loss.

3.1 AdvPC Attack Pipeline

3D Point Clouds ($\mathcal{X}$). We define a point cloud $\mathcal{X} \in \mathbb{R}^{N\times 3}$, as a set of N 3D points, where each point $\mathbf{x}_i \in \mathbb{R}^3$ is represented by its 3D coordinates (x_i, y_i, z_i).
Point Cloud Networks (F). We focus on 3D point cloud classifiers with a feature max pooling layer as detailed in Eq (1), where h_{mlp} and h_{conv} are MLP and Convolutional (1×1 or edge) layers, respectively. This produces a K-class classifier $\mathbf{F}$.

$$\mathbf{F}(\mathcal{X}) = h_{\text{mlp}}(\max_{\mathbf{x}_i \in \mathcal{X}} \{h_{\text{conv}}(\mathbf{x}_i)\}) \tag{1}$$

Here, $\mathbf{F}: \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^K$ produces the logits layer of the classifier with size K. For our attacks, we take $\mathbf{F}$ to be one of the following widely used networks in the

literature: PointNet [21], PointNet++ [22] in single-scale form (SSG) and multi-scale form (MSG), and DGCNN [29]. Section 5.2 delves deep into the differences between them in terms of their sensitivities to adversarial perturbations.

Point Cloud Auto-Encoder (G). An AE learns a representation of the data and acts as an effective defense against adversarial attacks. It ideally projects a perturbed point cloud onto the natural manifold of inputs. Any AE architecture in point clouds can be used, but we select the one in [1] because of its simple structure and effectiveness in recovering from adversarial perturbation. The AE $\mathbf{G}$ consists of an encoding part, $\mathbf{g}_{\text{encode}} : \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^q$ (similar to Eq (1)), and an MLP decoder, $\mathbf{g}_{\text{mlp}} : \mathbb{R}^q \rightarrow \mathbb{R}^{N\times 3}$, to produce a point cloud. It can be described formally as: $\mathbf{G}(.) = \mathbf{g}_{\text{mlp}}\big(\mathbf{g}_{\text{encode}}(.)\big)$. We train the AE with the Chamfer loss as in [1] on the same data used to train $\mathbf{F}$, such that it can reliably encode and decode 3D point clouds. We freeze the AE weights during the optimization of the adversarial perturbation on the input. Since the AE learns how naturally occurring point clouds look like, the gradients updating the attack, which is also tasked to fool the reconstructed sample after the AE, actually become more dependent on the data and less on the victim network. The enhanced data dependency of our attack results in the success of our attacks on unseen transfer networks besides the success on the victim network. As such, the proposed composition allows the crafted attack to successfully attack the victim classifier, as well as, fool transfer classifiers that operate on a similar input data manifold.

3.2 AdvPC Attack Loss

Soft Constraint Loss. In AdvPC attacks, like the ones in Fig. 3, we focus solely on perturbations of the input. We modify each point $\mathbf{x}_i$ by a an addictive perturbation variable δ_i. Formally, we define the perturbed point set $\mathcal{X}' = \mathcal{X} + \boldsymbol{\Delta}$, where $\boldsymbol{\Delta} \in \mathbb{R}^{N\times 3}$ is the perturbation parameter we are optimizing for. Consequently, each pair $(\mathbf{x}_i, \mathbf{x}'_i)$ are in correspondence. Adversarial attacks are commonly formulated as in Eq (2), where the goal is to find an input perturbation $\boldsymbol{\Delta}$ that successfully fools $\mathbf{F}$ into predicting an incorrect label t', while keeping $\mathcal{X}'$ and $\mathcal{X}$ close under distance metric $\mathcal{D}$: $\mathbb{R}^{N\times 3} \times \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}$.

$$\min_{\boldsymbol{\Delta}} \quad \mathcal{D}\left(\mathcal{X}, \mathcal{X}'\right) \qquad \text{s.t.} \quad \left[\arg\max_i \ \mathbf{F}\left(\mathcal{X}'\right)_i\right] = t' \tag{2}$$

The formulation in Eq (2) can describe targeted attacks (if t' is specified before the attack) or untargeted attacks (if t' is any label other than the true label of $\mathcal{X}$). We adopt the following choice of t' for untargeted attacks: $t' = \left[\arg\max_{i\neq \text{true}} \ \mathbf{F}\left(\mathcal{X}'\right)_i\right]$. Unless stated otherwise, we primarily use untargeted attacks in this paper. As pointed out in [4], it is difficult to directly solve Eq (2). Instead, previous works like [31,25] have used the well-known C&W formulation, giving rise to the commonly known soft constraint attack: $\min_{\boldsymbol{\Delta}} \ f_{t'}\left(\mathbf{F}(\mathcal{X}')\right) + \lambda \mathcal{D}\left(\mathcal{X}, \mathcal{X}'\right)$ where $f_{t'}\left(\mathbf{F}(\mathcal{X}')\right)$ is the adversarial loss function defined on the network $\mathbf{F}$ to move it to label t' as in Eq (3).

$$f_{t'}\left(\mathbf{F}(\mathcal{X}')\right) = \max\left(\max_{i\neq t'}\left(\mathbf{F}\left(\mathcal{X}'\right)_i\right) - \mathbf{F}\left(\mathcal{X}'\right)_{t'} + \kappa, 0\right), \tag{3}$$

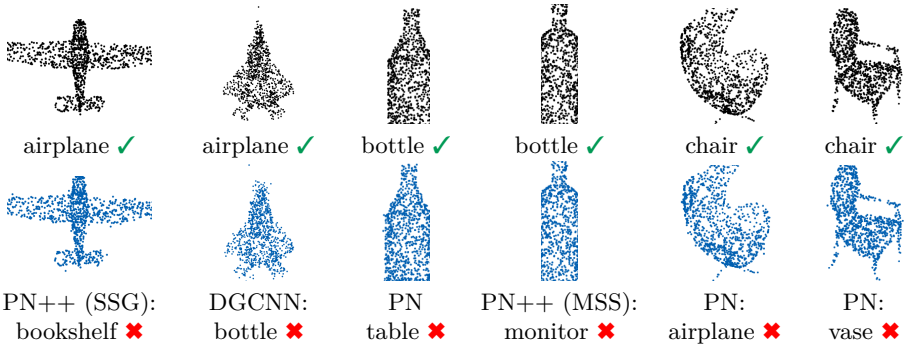

Fig. 3: **Examples of AdvPC Attacks:** Adversarial attacks are generated for victim networks PointNet, PointNet ++ (MSG/SSG) and DGCNN using AdvPC. The unperturbed point clouds are in black (*top*) while the perturbed examples are in blue (*bottom*). The network predictions are shown under each point cloud. The wrong prediction of each perturbed point cloud matches the target of the AdvPC attack.

where κ is a loss margin. The 3D-Adv attack [31] uses ℓ_2 for $\mathcal{D}(\mathcal{X}, \mathcal{X}')$, while the KNN Attack [25] uses Chamfer Distance.

Hard Constraint Loss. An alternative to Eq (2) is to put $\mathcal{D}(\mathcal{X}, \mathcal{X}')$ as a hard constraint, where the objective can be minimized using Projected Gradient Descent (PGD) [11,16] as follows.

$$\min_{\boldsymbol{\Delta}} \quad f_{t'}\left(\mathbf{F}(\mathcal{X}')\right) \qquad s.t. \;\; \mathcal{D}(\mathcal{X}, \mathcal{X}') \leq \epsilon \tag{4}$$

Using a hard constraint sets a limit to the amount of added perturbation in the attack. This limit is defined by ϵ in Eq (4), which we call norm-budget in this work. Having this bound ensures a fair comparison between different attack schemes. We compare these schemes by measuring their attack success rate at different levels of norm-budget. Using PGD, the above optimization in Eq (4) with ℓ_p distance $\mathcal{D}_{\ell_p}(\mathcal{X}, \mathcal{X}')$ can be solved by iteratively projecting the perturbation $\boldsymbol{\Delta}$ onto the ℓ_p sphere of size ϵ_p after each gradient step such that: $\boldsymbol{\Delta}_{t+1} = \Pi_p\left(\boldsymbol{\Delta}_t - \eta \nabla_{\boldsymbol{\Delta}_t} f_{t'}\left(\mathbf{F}(\mathcal{X}')\right), \epsilon_p\right)$. Here, $\Pi_p\left(\boldsymbol{\Delta}, \epsilon_p\right)$ projects the perturbation $\boldsymbol{\Delta}$ onto the ℓ_p sphere of size ϵ_p, and η is a step size. The two most commonly used ℓ_p distance metrics in the literature are ℓ_2, which measures the energy of the perturbation, and ℓ_∞, which measures the maximum point perturbation of each $\boldsymbol{\delta}_i \in \boldsymbol{\Delta}$. In our experiments, we choose to use the ℓ_∞ distance defined as $\mathcal{D}_{\ell_\infty}(\mathcal{X}, \mathcal{X}') = \max_i \|\boldsymbol{\delta}_i\|_\infty$, The projection of $\boldsymbol{\Delta}$ onto the ℓ_∞ sphere of size ϵ_∞ is: $\Pi_\infty\left(\boldsymbol{\Delta}, \epsilon_\infty\right) = \mathrm{SAT}_{\epsilon_\infty}(\boldsymbol{\delta}_i), \; \forall \boldsymbol{\delta}_i \in \boldsymbol{\Delta}$, where $\mathrm{SAT}_{\epsilon_\infty}(\boldsymbol{\delta}_i)$ is the element-wise saturation function that takes every element of vector $\boldsymbol{\delta}_i$ and limits its range to $[-\epsilon_\infty, \epsilon_\infty]$. Norm-budget ϵ_∞ is used throughout the experiments in this work.

In **supplement**, we detail our formulation when ℓ_2 is used as the distance metric and report similar superiority over the baselines just as the ℓ_∞ results. For completeness, we also show in the supplement the effect of using different

distance metrics (ℓ_2, Chamfer, and Earth Mover Distance) as soft constraints on transferability and attack effectiveness.

Data Adversarial Loss. The objectives in Eq (2, 4) focus solely on the network $\mathbf{F}$. We also want to add more focus on the data in crafting our attacks. We do so by fooling $\mathbf{F}$ using both the perturbed input $\mathcal{X}'$ and the AE reconstruction $\mathbf{G}(\mathcal{X}')$ (see Fig. 2). Our new objective becomes:

$$\min_{\boldsymbol{\Delta}} \quad \mathcal{D}\left(\mathcal{X}, \mathcal{X}'\right) \qquad \text{s.t. } [\arg\max_i \ \mathbf{F}\left(\mathcal{X}'\right)_i] = t'; \quad [\arg\max_i \ \mathbf{F}\left(\mathbf{G}(\mathcal{X}')\right)_i] = t'' \tag{5}$$

Here, t'' is any incorrect label $t'' \neq \arg\max_i \mathbf{F}\left(\mathcal{X}\right)_i$ and t' is just like Eq (2). The second constraint ensures that the prediction of the perturbed sample after the AE differs from the true label of the unperturbed sample. Similar to Eq (2), this objective is hard to optimize, so we follow similar steps as in Eq (4) and optimize the following objective for AdvPC using PGD (with ℓ_∞ as the distance metric):

$$\min_{\boldsymbol{\Delta}} \ (1-\gamma)\ f_{t'}\left(\mathbf{F}(\mathcal{X}')\right) + \gamma\ f_{t''}\left(\mathbf{F}\left(\mathbf{G}(\mathcal{X}')\right)\right) \quad s.t. \ \ \mathcal{D}_{\ell_\infty}\left(\mathcal{X}, \mathcal{X}'\right) \leq \epsilon_\infty \tag{6}$$

Here, f is as in Eq (3), while γ is a hyper-parameter that trades off the attack's success before and after the AE . When $\gamma = 0$, the formulation in Eq (6) becomes Eq (4). We use PGD to solve Eq (6) just like Eq (4). We follow the same procedures as in [31] when solving Eq (6) by keeping a record of any $\boldsymbol{\Delta}$ that satisfies the constraints in Eq (5) and by trying different initializations for $\boldsymbol{\Delta}$.

4 Experiments

4.1 Setup

Dataset and Networks. We use ModelNet40 [30] to train the classifier network ($\mathbf{F}$) and the AE network ($\mathbf{G}$), as well as test our attacks. ModelNet40 contains 12,311 CAD models from 40 different classes. These models are divided into 9,843 for training and 2,468 for testing. Similar to previous work [38,31,37], we sample 1,024 points from each object. We train the $\mathbf{F}$ victim networks: PointNet[21], PointNet++ in both Single-Scale (SSG) and Multi-scale (MSG) [22] settings, and DGCNN [29]. For a fair comparison, we adopt the subset of ModelNet40 detailed in [31] to perform and evaluate our attacks against their work (we call this the attack set). In the attack set, 250 examples are chosen from 10 ModelNet40 classes. We train the AE using the full ModelNet40 training set with the Chamfer Distance loss and then fix the AE when the attacks are being generated.

Adversarial Attack Methods. We compare AdvPC against the state-of-the-art baselines 3D-Adv [31] and KNN Attack [25]. For all attacks, we use Adam optimizer [10] with learning rate $\eta = 0.01$, and perform 2 different initializations for the optimization of $\boldsymbol{\Delta}$ (as done in [31]). The number of iterations for the attack optimization for all the networks is 200. We set the loss margin $\kappa = 30$ in Eq (3) for both 3D-Adv [31] and AdvPC and $\kappa = 15$ for KNN Attack [25] (as suggested in their paper). For other hyperparameters of [31,25], we follow what is reported in their papers. We pick $\gamma = 0.25$ in Eq (6) for AdvPC because it

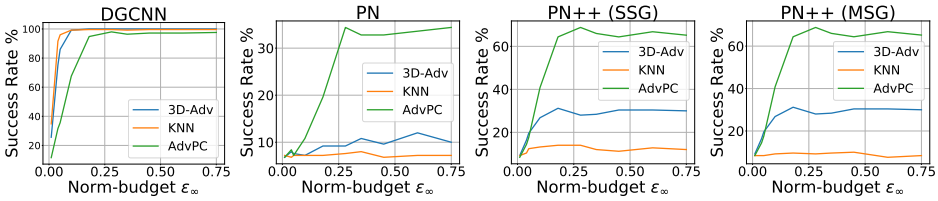

Fig. 4: **Transferability Across Different Norm-Budgets**: Here, the victim network is DGCNN [29] and the attacks are optimized using different ϵ_∞ norm-budgets. We report the attack success on DGCNN and on the transfer networks (PointNet, PointNet ++ MSG, and PointNet++ SSG). We note that our AdvPC transfers better to the other networks across different ϵ_∞ as compared to the baselines 3D-Adv[31] and KNN Attack [25]. Similar plots for the other victim networks are provided in the **supplement**.

strikes a balance between the success of the attack and its transferability (refer to Section 5.1 for details). In all of the attacks, we follow the same procedure as [31], where the best attack that satisfies the objective during the optimization is reported. We add the hard ℓ_∞ projection $\Pi_\infty(\boldsymbol{\Delta}, \epsilon_\infty)$ described in Section 3 to all the methods to ensure fair comparison on the same norm-budget ϵ_∞. We report the best performance of the baselines obtained under this setup.

Transferability. We follow the same setup as [19,20] by generating attacks using the constrained ℓ_∞ metric and measure their success rate at different norm-budgets ϵ_∞ taken to be in the range $[0, 0.75]$. This range is chosen because it enables the attacks to reach 100% success on the victim network, as well as offer an opportunity for transferability to other networks. We compare AdvPC against the state-of-the-art baselines [31,25] under these norm-budgets (*e.g.* see Fig. 4 for attacking DGCNN). To measure the success of the attack, we compute the percentage of samples out of all attacked samples that the victim network misclassified. We also measure transferability from each victim network to the transfer networks. For each pair of networks, we optimize the attack on one network (victim) and measure the success rate of this optimized attack when applied as input to the other network (transfer). We report these success rates for all network pairs. No defenses are used in the transferability experiment. All the attacks performed in this section are untargeted attacks (following the convention for transferability experiments [31]).

Attacking the Defenses. We also analyze the success of our attacks against point cloud defenses. We compare AdvPC attacks and the baselines [31,25] against several defenses used in the point cloud literature: SOR, SRS, DUP-Net [38], and Adversarial Training [31]. We also add a newly trained AE (different from the one used in the AdvPC attack) to this list of defenses. For SRS, we use a drop rate of 10%, while in SOR, we use the same parameters proposed in [38]. We train DUP-Net on ModelNet40 with an up-sampling rate of 2. For Adversarial Training, all four networks are trained using a mix of the training data of ModelNet40 and adversarial attacks generated by [31]. While these experiments are for untargeted

- Victim Network	- Attack	$\epsilon_\infty = 0.18$ PN	PN++ (MSG)	PN++ (SSG)	DGCNN	$\epsilon_\infty = 0.45$ PN	PN++ (MSG)	PN++ (SSG)	DGCNN
PN	3D-Adv [31]	100	8.4	10.4	6.8	100	8.8	9.6	8.0
	KNN [25]	100	9.6	10.8	6.0	100	9.6	8.4	6.4
	AdvPC (Ours)	98.8	**20.4**	**27.6**	**22.4**	98.8	**18.0**	**26.8**	**20.4**
PN++ (MSG)	3D-Adv [31]	6.8	100	28.4	11.2	7.2	100	29.2	11.2
	KNN [25]	6.4	100	22.0	8.8	6.4	100	23.2	7.6
	AdvPC (Ours)	**13.2**	97.2	**54.8**	**39.6**	**18.4**	98.0	**58.0**	**39.2**
PN++ (SSG)	3D-Adv [31]	7.6	9.6	100	6.0	7.2	10.4	100	7.2
	KNN [25]	6.4	9.2	100	6.4	6.8	7.6	100	6.0
	AdvPC (Ours)	**12.0**	**27.2**	99.2	**22.8**	**14.0**	**30.8**	99.2	**27.6**
DGCNN	3D-Adv [31]	9.2	11.2	31.2	100	9.6	12.8	30.4	100
	KNN [25]	7.2	9.6	14.0	99.6	6.8	10.0	11.2	99.6
	AdvPC (Ours)	**19.6**	**46.0**	**64.4**	94.8	**32.8**	**48.8**	**64.4**	97.2

Table 1: **Transferability of Attacks**: We use norm-budgets (max ℓ_∞ norm allowed in the perturbation) of $\epsilon_\infty = 0.18$ and $\epsilon_\infty = 0.45$. All the reported results are the untargeted Attack Success Rate (higher numbers are better attacks). **Bold** numbers indicate the most transferable attacks. Our attack consistently achieves better transferability than the other attacks for all networks, especially on DGCNN [29]. For reference, the classification accuracies on unperturbed samples for networks PN, PN++(MSG), PN++(SSG) and DGCNN are 92.8%, 91.5%, 91.5%, and 93.7%, respectively.

attacks, we perform similar experiments under targeted attacks and report the results in **supplement** for reference and completeness.

4.2 Results

We present quantitative results that focus on two main aspects. First, we show the transferable power of AdvPC attacks to different point cloud networks. Second, we highlight the strength of AdvPC under different point cloud defenses.

Transferability. Table 1 reports transferability results for $\epsilon_\infty = 0.18$ and $\epsilon_\infty = 0.45$ and compares AdvPC with the baselines [31,25]. The value $\epsilon_\infty = 0.18$ is chosen, since it allows the DGCNN attack to reach maximum success (see Section 5.2), and the value $\epsilon_\infty = 0.45$ is arbitrarily chosen to be midway in the remaining range of ϵ_∞. It is clear that AdvPC attacks consistently beat the baselines when transferring between networks (up to 40%). Our method shows substantial gains in the case of DGCNN. We also report transferability results for a range of ϵ_∞ values in Fig. 4 when the victim network is DGCNN, and the attacks transferred to all other networks. In **supplement**, we show the same plots when the victim network is taken to be PN and PN++. To represent all these transferability curves compactly, we aggregate their results into a Transferability Matrix. Every entry in this matrix measures the transferability from the victim network (**row**) to the transfer network (**column**), and it is computed as the

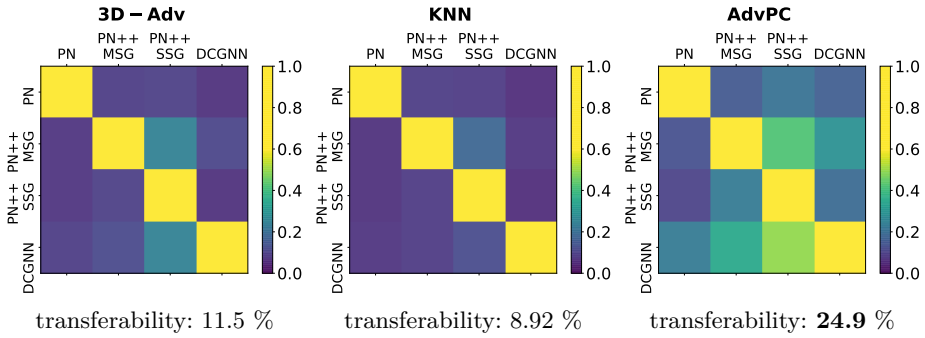

Fig. 5: **Transferability Matrix**: Visualizing the overall transferability for 3D-Adv [31] (*left*), KNN Attack [25] (*middle*), and our AdvPC (*right*). Elements in the same row correspond to the same victim network used in the attack, while those in the same column correspond to the network that the attack is transferred to. Each matrix element measures the average success rate over the range of ϵ_∞ for the transfer network. We expect the diagonal elements of each transferability matrix (average success rate on the victim network) to have high values, since each attack is optimized on the same network it is transferred to. More importantly, brighter off-diagonal matrix elements indicate better transferability. We observe that our proposed AdvPC attack is more transferable than the other attacks and that DGCNN is a more transferable victim network than the other point cloud networks. The transferability score under each matrix is the average of the off-diagonal matrix values, which summarizes overall transferability for an attack.

average success rate of the attack evaluated on the transfer network across all ϵ_∞ values. This value reflects how good the perturbation is at fooling the transfer network overall. As such, we advocate the use of the transferability matrix as a standard mode of evaluation for future work on network-transferable attacks. In Fig. 5, we show the transferability matrices for our attack and the baselines. AdvPC transfers better overall, since it leads to higher (brighter) off-diagonal values in the matrix. Using the average of off-diagonal elements in this matrix as a single scalar measure of transferability, AdvPC achieves 24.9% average transferability, as compared to 11.5% for 3D-Adv [31] and 8.92% for KNN Attack [25]. We note that DGCNN [29] performs best in terms of transferability and is the hardest network to attack (for AdvPC and the baselines).

Attacking Defenses. Since DGCNN performs the best in transferability, we use it to evaluate the resilience of our AdvPC attacks under different defenses. We use the five defenses described in Section 4.1 and report their results in Table 2. Our attack is more resilient than the baselines against all defenses. We note that the AE defense is very strong against all attacks compared to other defenses [38], which explains why AdvPC works very well against other defenses and transfers well to unseen networks. We also observe that our attack is strong against simple statistical defenses like **SRS** (38% improvement over the baselines). We report results for other victim networks (PN and PN++) in the **supplement** , where AdvPC shows superior performance against the baselines under these defenses.

Defenses	$\epsilon_\infty = 0.18$			$\epsilon_\infty = 0.45$		
	3D-Adv [31]	**KNN [25]**	**AdvPC (ours)**	**3D-Adv [31]**	**KNN [25]**	**AdvPC (ours)**
No defense	**100**	99.6	94.8	**100**	99.6	97.2
AE (newly trained)	9.2	10.0	**17.2**	12.0	10.0	**21.2**
Adv Training [31]	7.2	7.6	**39.6**	8.8	7.2	**42.4**
SOR [38]	18.8	17.2	**36.8**	19.2	19.2	**32.0**
DUP Net [38]	28	28.8	**43.6**	28	31.2	**37.2**
SRS [38]	43.2	29.2	**80.0**	47.6	31.2	**85.6**

Table 2: **Attacking Point Cloud Defenses:** We evaluate untargeted attacks using norm-budgets of $\epsilon_\infty = 0.18$ and $\epsilon_\infty = 0.45$ with DGCNN [29] as the victim network under different defenses for 3D point clouds. Similar to before, we report attack success rates (**higher** indicates better attack). AdvPC consistently outperforms the other attacks [31,25] for all defenses. Note that both the attacks *and* evaluations are performed on DGCNN, which has an accuracy of 93.7% without input perturbations (for reference).

5 Analysis

We perform several analytical experiments to further explore the results obtained in Section 4.2. We first study the effect of different factors that play a role in the transferability of our attacks. We also show some interesting insights related to the sensitivity of point cloud networks and the effect of the AE on the attacks.

5.1 Ablation Study (hyperparameter γ)

Here, we study the effect of γ used in Eq (6) on the performance of our attacks. While varying γ between 0 and 1, we record the attack success rate on the victim network and report the transferability to all of the other three transfer networks (average success rate on the transfer networks). We present averaged results over all norm-budgets in Fig. 6 for the four victim networks. One observation is that adding the AE loss with $\gamma > 0$ tends to deteriorate the success rate, even though it improves transferability. We pick $\gamma = 0.25$ in our experiments to balance success and transferability.

5.2 Network Sensitivity to Point Cloud Attacks

Fig. 7 plots the sensitivity of the various networks when they are subject to input perturbations of varying norm-budgets ϵ_∞. We measure the classification accuracy of each network under our AdvPC attack ($\gamma = 0.25$), 3D-Adv [31], and KNN Attack [25]. We observe that DGCNN [29] tends to be the most robust to adversarial perturbations in general. This might be explained by the fact that the convolution neighborhoods in DGCNN are dynamically updated across layers and iterations. This dynamic behavior in network structure may hinder the effect of the attack because gradient directions can change significantly from one iteration to another. This leads to failing attacks and higher robustness for DGCNN [29].

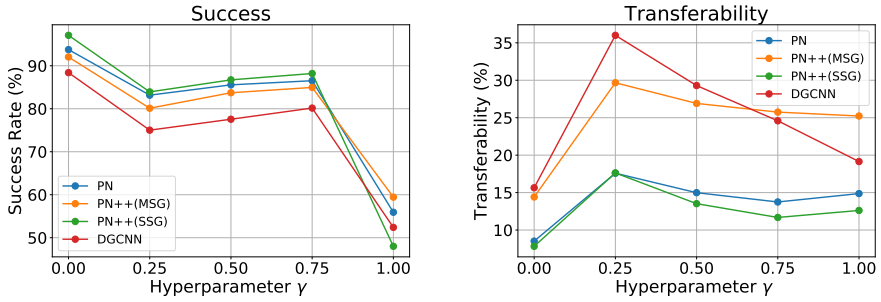

Fig. 6: **Ablation Study**: Studying the effect of changing AdvPC hyperparameter (γ) on the success rate of the attack (*left*) and on its transferability (*right*). The transferability score reported for each victim network is the average success rate on the transfer networks averaged across all different norm-budgets ϵ_∞. We note that as γ increases, the success rate of the attack on the victim network drops, and the transferability varies with γ. We pick $\gamma = 0.25$ in all of our experiments.

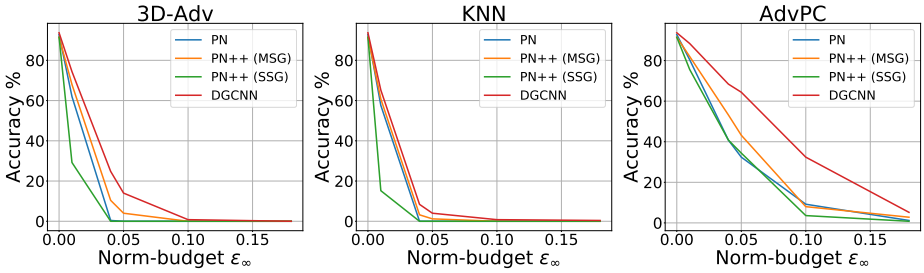

Fig. 7: **Sensitivity of Architectures**: We evaluate the sensitivity of each of the four networks for increasing norm-budget. For each network, we plot the classification accuracy under 3D-Adv perturbation [31] (*left*), KNN Attack [25] (*middle*), and our AdvPC attack (*right*). Overall, DGCNN [29] is affected the least by adversarial perturbation.

5.3 Effect of the Auto-Encoder (AE)

In Fig. 8, we show an example of how AE reconstruction preserves the details of the unperturbed point cloud and does not change the classifier prediction. When a perturbed point cloud passes through the AE, it recovers a natural-looking shape. The AE's ability to reconstruct natural-looking 3D point clouds from various perturbed inputs might explain why it is a strong defense against attacks in Table 2. Another observation from Fig. 8 is that: when we fix the target t' and do not enforce a specific incorrect target t'' (*i.e.* untargeted attack setting) for the data adversarial loss on the reconstructed point cloud in the AdvPC attack (Eq (6)), the optimization mechanism tends to pick t'' to be a *similar* class to the correct one. For example, a *Toilet* point cloud perturbed by AdvPC can be transformed into a *Chair* (similar in appearance to a toilet), if reconstructed by the AE. This effect is not observed for the other attacks [31,25], which do not consider the

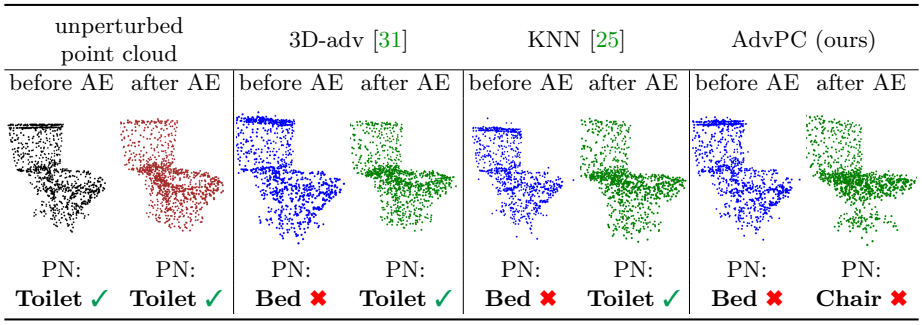

Fig. 8: **Effect of the Auto-Encoder (AE):** The AE does not affect the unperturbed point cloud (classified correctly by PN before and after AE). The AE cleans the point cloud perturbed by 3D-Adv and KNN [31,25], which allows PN to predict the correct class label. However, our AdvPC attack can fool PN before and after AE reconstruction. Samples perturbed by AdvPC, if passed through the AE, transform into similar looking objects from different classes (Chair looks similar to Toilet).

data distribution and optimize solely for the network. For completeness, we tried replacing the AE with other 3D generative models from [1] in our AdvPC attack, and we tried to use the learning approach in [19,20] instead of optimization, but the attack success was less than satisfactory in both cases (refer to **supplement**).

6 Conclusions

In this paper, we propose a new adversarial attack for 3D point clouds that utilizes a data adversarial loss to formulate network-transferable perturbations. Our attacks achieve better transferability to four popular point cloud networks than other 3D attacks, and they improve robustness against popular defenses. Future work would extend this attack to other 3D deep learning tasks, such as detection and segmentation, and integrate it into a robust training framework for point cloud networks.

Acknowledgments. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research under Award No. RGC/3/3570-01-01.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3d point clouds. International Conference on Machine Learning (ICML) (2018)
2. Alcorn, M.A., Li, Q., Gong, Z., Wang, C., Mai, L., Ku, W.S., Nguyen, A.: Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

3. Cao, Y., Xiao, C., Yang, D., Fang, J., Yang, R., Liu, M., Li, B.: Adversarial objects against lidar-based autonomous driving systems. CoRR **abs/1907.05418** (2019)
4. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: IEEE Symposium on Security and Privacy (SP) (2017)
5. Engelmann, F., Kontogianni, T., Hermans, A., Leibe, B.: Exploring spatial context for 3d semantic segmentation of point clouds. In: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW). pp. 716–724 (Oct 2017)
6. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (ICLR) (2015)
7. Hamdi, A., Ghanem, B.: Towards analyzing semantic robustness of deep neural networks. CoRR **abs/1904.04621** (2019)
8. Hamdi, A., Muller, M., Ghanem, B.: SADA: semantic adversarial diagnostic attacks for autonomous applications. In: AAAI Conference on Artificial Intelligence (2020)
9. Huang, Q., Wang, W., Neumann, U.: Recurrent slice networks for 3d segmentation of point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2626–2635 (2018)
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
11. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial machine learning at scale. CoRR **abs/1611.01236** (2016)
12. Landrieu, L., Boussaha, M.: Point cloud oversegmentation with graph-structured deep metric learning pp. 7440–7449 (2019)
13. Landrieu, L., Simonovsky, M.: Large-scale point cloud semantic segmentation with superpoint graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4558–4567 (2018)
14. Li, J., Chen, B.M., Hee Lee, G.: So-net: Self-organizing network for point cloud analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9397–9406 (2018)
15. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. In: Advances in neural information processing systems (NIPS). pp. 820–830 (2018)
16. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: International Conference on Learning Representations (ICLR) (2018)
17. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
18. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: A simple and accurate method to fool deep neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
19. Naseer, M.M., Khan, S.H., Khan, M.H., Shahbaz Khan, F., Porikli, F.: Cross-domain transferability of adversarial perturbations. In: Advances in Neural Information Processing Systems (NeurIPS), pp. 12905–12915 (2019)
20. Poursaeed, O., Katsman, I., Gao, B., Belongie, S.: Generative adversarial perturbations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4422–4431 (2018)
21. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 652–660 (2017)

22. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems (NIPS). pp. 5099–5108 (2017)
23. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. CoRR **abs/1312.6199** (2013)
24. Tatarchenko, M., Park, J., Koltun, V., Zhou, Q.Y.: Tangent convolutions for dense prediction in 3d. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3887–3896 (2018)
25. Tsai, T., Yang, K., Ho, T.Y., Jin, Y.: Robust adversarial objects against deep learning models. In: AAAI Conference on Artificial Intelligence (2020)
26. Tu, C.C., Ting, P., Chen, P.Y., Liu, S., Zhang, H., Yi, J., Hsieh, C.J., Cheng, S.M.: Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 742–749 (2019)
27. Tu, J., Ren, M., Manivasagam, S., Liang, M., Yang, B., Du, R., Cheng, F., Urtasun, R.: Physically realizable adversarial examples for lidar object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13716–13725 (2020)
28. Wang, W., Yu, R., Huang, Q., Neumann, U.: Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2569–2578 (2018)
29. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics (TOG) (2019)
30. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1912–1920 (2015)
31. Xiang, C., Qi, C.R., Li, B.: Generating 3d adversarial point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9136–9144 (2019)
32. Xiao, C., Yang, D., Li, B., Deng, J., Liu, M.: Meshadv: Adversarial meshes for visual recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6898–6907 (2019)
33. Ye, X., Li, J., Huang, H., Du, L., Zhang, X.: 3d recurrent neural networks with context fusion for point cloud semantic segmentation. In: European Conference on Computer Vision (ECCV). pp. 415–430. Springer (2018)
34. Yu, L., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: Pu-net: Point cloud upsampling network. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
35. Zeng, X., Liu, C., Wang, Y.S., Qiu, W., Xie, L., Tai, Y.W., Tang, C.K., Yuille, A.L.: Adversarial attacks beyond the image space. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
36. Zhao, Z., Dua, D., Singh, S.: Generating natural adversarial examples. In: International Conference on Learning Representations (ICLR) (2018)
37. Zheng, T., Chen, C., Yuan, J., Li, B., Ren, K.: Pointcloud saliency maps. In: The IEEE International Conference on Computer Vision (ICCV) (2019)
38. Zhou, H., Chen, K., Zhang, W., Fang, H., Zhou, W., Yu, N.: Dup-net: Denoiser and upsampler network for 3d adversarial point clouds defense. In: The IEEE International Conference on Computer Vision (ICCV) (2019)