AdvPC: Transferable Adversarial Perturbations on 3D Point Clouds Supplementary Material

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

1 Background on Point Cloud Distances

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). In this work, we focus solely on the perturbations of the input. This means we modify each point $\mathbf{x}_i$ by a perturbation variable. 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.

1.1 Trivial Distances (ℓ_p)

The most commonly used distance metric in adversarial attacks in the image domain is ℓ_p. Unlike image domain where every pixel corresponds to the perturbed pixel in adversarial attacks, in point clouds adversarial attacks by adding, removing, or transforming the point cloud destroys the correspondence relationship to the unperturbed sample. Hence, it becomes infeasible to accurately calculate the ℓ_p metric for the attack. In our paper, we focus on adversarial perturbations, which preserves the matching between the unperturbed sample and the perturbed sample. This property of preservation of matching points allows us to measure the ℓ_p norms of the attack exactly, which allow for standard evaluation similar to the one in the image domain.Here we assume the two point-sets are equal in size and are aligned , *i.e.* for $\mathbf{x}_i \in \mathcal{X}$, $\mathbf{x}'_i = \mathbf{x}_i + \boldsymbol{\delta}_i$, $i \in 1, 2, ..., N$

$$\mathcal{D}_{\ell_p}(\mathcal{X}, \mathcal{X}') = \left(\sum_i \|\boldsymbol{\delta}_i\|_p^p \right)^{\frac{1}{p}} \tag{1}$$

For our attacks, we use the ℓ_2 and ℓ_∞ distances, defined in (2) and (3) respectively. The ℓ_2 distance measures the energy of the perturbation, while ℓ_∞ represents the maximum allowed perturbation of each $\delta_i \in \boldsymbol{\Delta}$.

ℓ_2 distance,. The ℓ_2 measures the energy of the perturbation performed on the unperturbed point cloud . Its calculation is similar to calculating the Frobenius norm of the matrix $\mathbf{X}$ that represent the point set perturbation variable $\boldsymbol{\Delta}$ such

that each row of $\mathbf{X}$ is a point $\delta_i \in \boldsymbol{\Delta}$. The ℓ_2 distance between two point sets can be measured as follows

$$\mathcal{D}_{\ell_2}(\mathcal{X}, \mathcal{X}') = \left(\sum_i \|\boldsymbol{\delta}_i\|_2^2 \right)^{\frac{1}{2}} = \|\boldsymbol{\Delta}\|_{\mathrm{F}} \tag{2}$$

ℓ_∞ distance,. The ℓ_∞ represents the max allowed perturbation at any dimension to every single point δ_i in the perturbation set $\boldsymbol{\Delta}$. This distance between two point sets can be measured as follows :

$$\mathcal{D}_{\ell_\infty}(\mathcal{X}, \mathcal{X}') = \max_i \|\boldsymbol{\delta}_i\|_\infty \tag{3}$$

1.2 Non-trivial Distances

Other point cloud distances that are commonly used in the literature do not require the two sets to be in a known correspondence (like the strict ℓ_p). These distance metrics include the following: Chamfer Distances, Hausdorff Distance, and Earth Mover Distance. In what follows, we formally present each of these metrics.

Chamfer Distance (CD). This is a common distance to compare 2 point sets. CD measures the average distance between closest point pairs of 2 different point clouds. We define CD in Eq (4).

$$\mathcal{D}_{CD}(\mathcal{X}, \mathcal{X}') = \frac{1}{\|\mathcal{X}'\|_0} \sum_{\mathbf{x}'_i \in \mathcal{X}'} \min_{\mathbf{x}_i \in \mathcal{X}} \|\mathbf{x}_i - \mathbf{x}'_i\|_2^2 \tag{4}$$

Hausdorff distance (HD). With HD, we compute the largest distance in the set of containing $\mathbf{x} \in \mathcal{X}$ and its closest point $\mathbf{x}' \in \mathcal{X}'$. We define HD as follows:

$$\mathcal{D}_H(\mathcal{X}, \mathcal{X}') = \max_{\mathbf{x}'_i \in \mathcal{X}'} \min_{\mathbf{x}_i \in \mathcal{X}} \|\mathbf{x}_i - \mathbf{x}'_i\|_2^2 \tag{5}$$

Earth Mover Distance (EMD). The EMD measures the total *effort* performed in the optimal transport scheme that transforms the first point set to the other. It is defined as follows:

$$\mathcal{D}_{EMD}(\mathcal{X}, \mathcal{X}') = \min_{\phi: \mathcal{X} \to \mathcal{X}'} \sum_i \|\mathbf{x}'_i - \phi(\mathbf{x}_i)\|_2, \tag{6}$$

where $\phi : \mathcal{X} \to \mathcal{X}'$ is a bijection transform.

2 Our Full Formulation

The pipeline of AdvPC is illustrated in Fig. 1. It consists of an Auto-Encoder (AE) **G**, which is trained to reconstruct 3D point clouds, and a point cloud classifier **F**. We seek to find a perturbation variable $\boldsymbol{\Delta}$ added to the input $\mathcal{X}$ to fool **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 (leveraged by the AE). 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, and then we present our attack setup and loss.

2.1 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 (7), where h_{mlp} and h_{conv} are MLP and Convolutional (1×1 or edge) layers respectively. This produces a K-class classifier **F**.

$$\mathbf{F}(\mathcal{X}) = h_{\text{mlp}}(\max_{\mathbf{x}_i \in \mathcal{X}} \{h_{\text{conv}}(\mathbf{x}_i)\}) \tag{7}$$

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 **F** to be one of the following widely used networks in the literature: PointNet [8], PointNet++ [9] in single-scale form (SSG) and multi-scale form (MSG), and DGCNN [11]. Section 7.2 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 in our pipeline, but we select the one in [1] because of its simple structure and effectiveness in recovering from adversarial perturbation. The AE **G** consists of an encoding part, $\mathbf{g}_{\text{encode}}: \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^q$ (similar to Eq (7)), 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 **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. We show in Section 6 how the AE acts as an effective defense against previous point cloud adversarial perturbations. 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

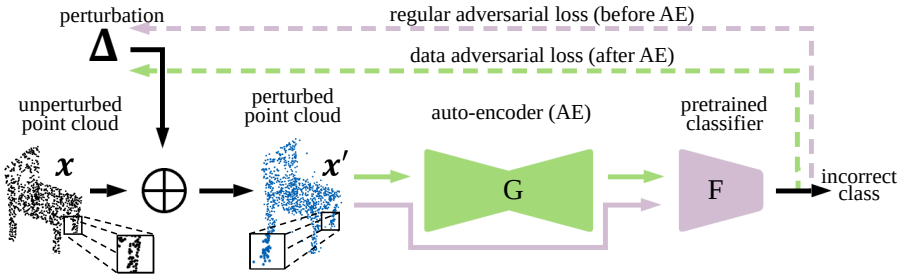

Fig. 1: **AdvPC Attack Pipeline:** We optimize for the constrained perturbation variable $\boldsymbol{\Delta}$ to generate the perturbed sample $\mathcal{X}' = \mathcal{X} + \boldsymbol{\Delta}$. The perturbed sample fools a trained classifier $\mathbf{F}$ (*i.e.* $\mathbf{F}(\mathcal{X}')$ is incorrect), 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 (16) and has two parts: network adversarial loss (*purple*) and data adversarial loss (*green*). Dotted lines are gradients flowing to the perturbation variable $\boldsymbol{\Delta}$.

to successfully attack the victim classifier, as well as, fool transfer classifiers that operate on a similar input data manifold. Furthermore, since many of the available defenses rely on natural statistics of 3D point clouds [15], we show that attacking the classifier after AE reconstruction can also lead to perturbations resilient to these defenses.

2.2 AdvPC Attack Loss

Soft Constraint Loss. In AdvPC attacks, like the ones in Fig. 2, 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 (8), 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 $\mathcal{D}$: $\mathbb{R}^{N\times 3} \times \mathbb{R}^{N\times 3} \rightarrow \mathbb{R}$.

$$\min_{\boldsymbol{\Delta}} \quad \mathcal{D}\left(\mathcal{X}, \mathcal{X}'\right) \qquad \text{s.t.} \quad \left[\arg\max_{i} \ \mathbf{F}\left(\mathcal{X}'\right)_i\right] = t' \tag{8}$$

The formulation in Eq (8) 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: $t' = \left[\arg\max_{i\neq \text{true}} \ \mathbf{F}\left(\mathcal{X}'\right)_i\right]$.We present the results of both targeted and untargeted attacks in this supplementary. As pointed out in [2], it is difficult to directly solve Eq (8). Instead, previous works like [13,10] have used the well-known C&W formulation, giving rise to the commonly known soft constraint attack:

$$\min_{\boldsymbol{\Delta}} \quad f_{t'}\left(\mathbf{F}(\mathcal{X}')\right) + \ \lambda\, \mathcal{D}\left(\mathcal{X}, \mathcal{X}'\right) \tag{9}$$

where $\mathcal{D}(\mathcal{X}, \mathcal{X}')$ can be any of the distances proposed in Eq (2,4,6), while $f_{t'}(\mathbf{F}(\mathcal{X}'))$ is the targeted adversarial loss function defined on the network $\mathbf{F}$ to move it to target t' as in Eq (10).

$$f_{t'}(\mathbf{F}(\mathcal{X}')) = \max\left(\max_{i \neq t'}(\mathbf{F}(\mathcal{X}')_i) - \mathbf{F}(\mathcal{X}')_{t'} + \kappa, 0\right), \tag{10}$$

where κ is a loss margin. The 3D-Adv attack [13] uses ℓ_2 for $\mathcal{D}(\mathcal{X}, \mathcal{X}')$ while KNN attack [10] uses Chamfer Distance.

Hard Constraint Loss. An alternative to Eq (8) is to put $\mathcal{D}(\mathcal{X}, \mathcal{X}')$ as a hard constraint, where the objective can be minimized using Projected Gradient Descent (PGD) [4,5] as follows.

$$\min_{\boldsymbol{\Delta}} \quad f_{t'}(\mathbf{F}(\mathcal{X}')) \qquad s.t. \;\; \mathcal{D}(\mathcal{X}, \mathcal{X}') \leq \epsilon \tag{11}$$

Using a hard constraint sets a limit to the amount of added perturbation in the attack. This limit is defined by ϵ in Eq (11). Having this bound ensures fair comparisons between different attacks schemes. We can do these comparisons by measuring the effectiveness of these attacks at different levels of ϵ. Using PGD, the above optimization in Eq (11) with ℓ_p distance $\mathcal{D}_{\ell_p}(\mathcal{X}, \mathcal{X}')$ can be solved by iteratively projecting the perturbed sample onto the ℓ_p sphere of size ϵ_p such that:

$$\boldsymbol{\Delta}_{t+1} = \Pi_p\left(\boldsymbol{\Delta}_t - \eta \nabla_{\boldsymbol{\Delta}_t} f_{t'}(\mathbf{F}(\mathcal{X}')), \epsilon_p\right) \tag{12}$$

Here, $\Pi_p(\boldsymbol{\Delta}, \epsilon_p)$ 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 $\boldsymbol{\delta}_i \in \boldsymbol{\Delta}$. Our experiments use the ℓ_2 distance defined as in Eq (2). while the projection of $\boldsymbol{\Delta}$ onto the ℓ_2 sphere of size ϵ_2 is:

$$\Pi_2(\boldsymbol{\Delta}, \epsilon_2) = \frac{\epsilon_2}{\max\left(\|\boldsymbol{\Delta}\|_{\mathrm{F}}, \epsilon_2\right)} \boldsymbol{\Delta} \tag{13}$$

On the other hand, the ℓ_∞ projection formulation is as follows:

$$\Pi_\infty(\boldsymbol{\Delta}, \epsilon_\infty) = \mathrm{SAT}_{\epsilon_\infty}(\boldsymbol{\delta}_i), \quad \forall \boldsymbol{\delta}_i \in \boldsymbol{\Delta}, \tag{14}$$

here $\mathrm{SAT}_\zeta(\boldsymbol{\delta}_i)$ is the element-wise saturation function that takes every element of vector $\boldsymbol{\delta}_i$ and limit its range in $[-\zeta, \zeta]$.

Data Adversarial Loss. The objectives in Eq (8, 11) 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 $\mathbf{G}(\mathcal{X}')$. Our new objective becomes:

$$\min_{\boldsymbol{\Delta}} \quad \mathcal{D}(\mathcal{X}, \mathcal{X}') \qquad \text{s.t. } [\arg\max_i \; \mathbf{F}(\mathcal{X}')_i] = t'; \;\; [\arg\max_i \; \mathbf{F}(\mathbf{G}(\mathcal{X}'))_i] = t'' \tag{15}$$

Here, t'' is any incorrect label $t'' \neq \arg\max_i \mathbf{F}(\mathcal{X})_i$ and t' is just like Eq (8). 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 (8), this objective is hard to optimize, so we follow similar steps as in Eq (11) and optimize the following objective for AdvPC using PGD (using ℓ_p as the distance metric):

$$\min_{\boldsymbol{\Delta}} \; (1-\gamma)\; f_{t'}\left(\mathbf{F}(\mathcal{X}')\right) + \gamma\; f_{t''}\left(\mathbf{F}\left(\mathbf{G}(\mathcal{X}')\right)\right) \quad s.t. \;\; \mathcal{D}_{\ell_p}\left(\mathcal{X}, \mathcal{X}'\right) \leq \epsilon_p \tag{16}$$

Here,f is as in Eq (10), 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 (16) becomes Eq (11).. We use PGD to solve Eq (16) as follows.

$$\begin{aligned} \boldsymbol{\Delta}_{t+1} = \mathit{\Pi}_p\Big(\boldsymbol{\Delta}_t - \eta(1-\gamma)\nabla_{\boldsymbol{\Delta}_t}\; f_{t'}\big(\mathbf{F}(\mathcal{X}')\big) \\ - \eta\;\gamma\;\nabla_{\boldsymbol{\Delta}_t}\; f_{t''}\big(\mathbf{F}\left(\mathbf{G}(\mathcal{X}')\right)\big), \epsilon_p\Big) \end{aligned} \tag{17}$$

Where $\mathit{\Pi}_p$ is the projection to ℓ_p as in Eq (13,14)

We follow the same procedure as in [13] when solving the optimization in Eq (16) by keeping a record of any $\boldsymbol{\Delta}$ that satisfies the constraints in Eq (15) and by trying different initializations for $\boldsymbol{\Delta}$. If we achieve the constraints in Eq (15) in one of the optimizations' initializations, we try smaller hard norms in the following initialization in order to find a better solution (smaller norm). The is the exactly the Binary Search followed by [13] to find the best hyperparameter λ in Eq (9) that will result in the smallest norm perturbation that succeeds in the attack on that specific sample.

3 Qualitative Results

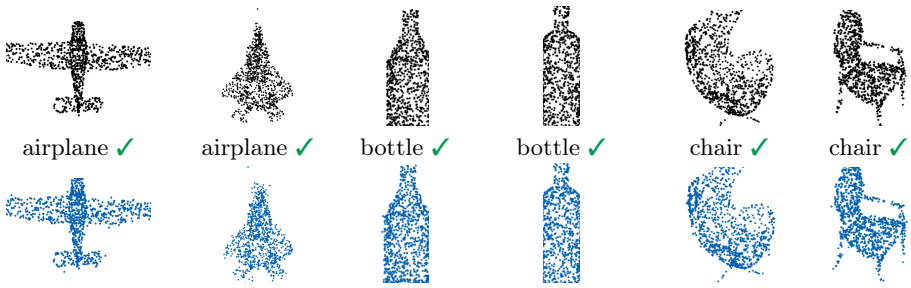

Fig. 2: **Examples of AdvPC Targeted Attacks:** Adversarial attacks are generated for victim networks PointNet, PointNet ++ (MSG/SSG) and DGCNN using AdvPC. The unperturbed point clouds are in black (*top*) while the perturbed examples are in blue (*bottom*). The network predictions are shown under each point cloud. The wrong prediction of each perturbed point cloud matches the target of the AdvPC attack.

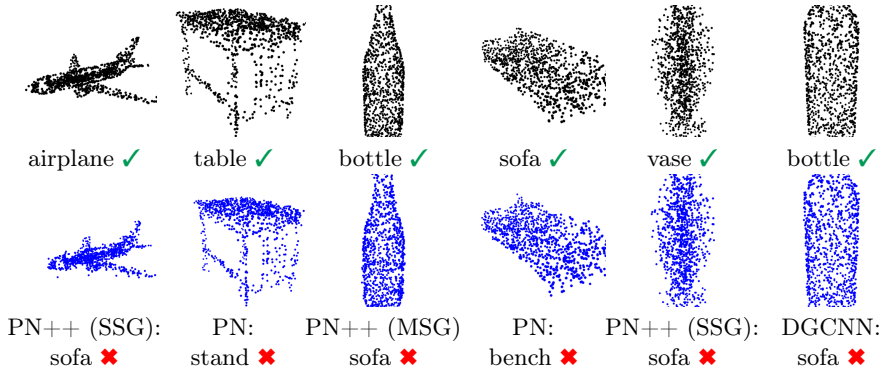

Fig. 3: **Examples of AdvPC Untargeted Attacks:** Adversarial attacks are generated for victim networks PointNet, PointNet ++ (MSG/SSG) and DGCNN using AdvPC. The unperturbed point clouds are in black (*top*) while the perturbed examples are in blue (*bottom*). The network predictions are shown under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

4 Experiments Setup

4.1 Dataset and Networks

We use ModelNet40 [12] to train the classifier network (**F**) and the AE network (**G**), as well as test our attacks. ModelNet40 contains 12,311 CAD models from 40 different classes. CAD models are divided into 9,843 for training and 2,468 for testing. Similar to previous work [15,13,14], we sample 1,024 points from each object. We train the **F** victim networks: PointNet[8], PointNet++ in both Single-Scale (SSG) and Multi-scale (MSG) [9] settings, and DGCNN [11]. For a fair comparison, we adopt the subset of ModelNet40 detailed in [13] 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. In untargeted attacks we only perform the attack once per test sample and report the average results. However, in targeted attacks (like the ones in Section 8) we evaluate the attacks on all the possible targets for each sample and report the average results as followed by c[13].

4.2 Adversarial Attack Methods

We compare AdvPC against the state-of-the-art baselines 3D-Adv [13] and KNN-Attack [10]. For all attacks, We use Adam optimizer [3] with learning rate $\eta = 0.01$, and perform 2 different initializations for the optimization of $\boldsymbol{\Delta}$ (as followed by [13]). The number of iterations for the attack optimization for all the networks is 200. We set the loss margin $\kappa = 30$ in Eq (10) for both 3D-Adv [13] and AdvPC and $\kappa = 15$ for KNN-Attack [10] (as they suggest in their paper). For other hyperparameters of [13,10], we follow what they report in their works. We pick $\gamma = 0.25$ in Eq (16) for AdvPC because it strikes a balance between the success of the attack and its transferability (refer to Section 7.1 for details). In all of the attacks, we follow the same procedure as [13], where the best attack that satisfies the objective during the optimization is reported. In this supplementary, we perform both the ℓ_∞ and ℓ_2 modes of attacks for the baselines and for AdvPC. For fair comparisons between the attack methods on the same norm-budgets, we add the following to all the attacks. For ℓ_∞ attacks, we add the hard projection $\Pi_\infty(\boldsymbol{\Delta}, \epsilon_\infty)$ (from Eq (14), while for ℓ_2 attack, we add the hard projection $\Pi_2(\boldsymbol{\Delta}, \epsilon_2)$ (from Eq (13). This insures that all the attacks have the same norm-budgets ϵ_∞ or ϵ_2 (depending on the attack mode).

4.3 Transferability

For the constrained ℓ_∞ metric, we measure their success rate at different norm-budgets ϵ_∞ taken to be in the range $[0, 0.75]$, whereas norm-budgets ϵ_2 is taken in the range $[0, 7]$. These ranges are chosen because they 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 [13,10] under these norm-budgets (*e.g.* see Fig. 5,4). The exact norms

used for ϵ_∞ and ϵ_2 are $\{0.01, 0.04, 0.05, 0.1, 0.18, 0.28, 0.35, 0.45, 0.6, 0.75\}$ and $\{0.1, 0.22, 0.48, 0.72, 1.0, 1.5, 1.8, 2.8, 4.0, 7.0\}$ respectively. At exactly $\epsilon_\infty = \epsilon_2 = 0$, we get the classification accuracies on unperturbed samples for networks PN, PN++(MSG), PN++(SSG) and DGCNN to be .92.8%, 91.5%, 91.5%, and 93.7% respectively. To measure the success of the attack, we measure 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 [13]). The results are reported in Sections 5.1,5.2, and 5.3

4.4 Attacking the Defenses

We also analyze the success of our attacks against point cloud defenses. We compare AdvPC attacks and the baselines [13,10] against several defenses used in the point cloud literature: SOR, SRS, DUP-Net [15], and Adversarial Training [13]. 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 [15]. 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 [13]. We always report the success rate as 1-accuracy of the victim networks on the perturbed data for that specific norm-budget. The results for untargeted attacks (ℓ_∞ and ℓ_2) are reported in Section 6, while for targeted attacks, the defense results (ℓ_∞ and ℓ_2) are reported in Section 8.

5 Full Transferability Results

5.1 Transferability on Specific Norms

- 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 [13]	100	8.4	10.4	6.8	100	8.8	9.6	8.0
	KNN [10]	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 [13]	6.8	100	28.4	11.2	7.2	100	29.2	11.2
	KNN [10]	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 [13]	7.6	9.6	100	6.0	7.2	10.4	100	7.2
	KNN [10]	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 [13]	9.2	11.2	31.2	100	9.6	12.8	30.4	100
	KNN [10]	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 under ℓ_∞ Norms**: 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 [11]. 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.

- Victim Network	- Attack	$\epsilon_2 = 1.8$ PN	PN++ (MSG)	PN++ (SSG)	DGCNN	$\epsilon_2 = 4.0$ PN	PN++ (MSG)	PN++ (SSG)	DGCNN
PN	3D-Adv [13]	100	8.4	8.8	7.2	100	7.6	9.6	6.0
	KNN [10]	100	9.2	8.4	7.2	100	8.8	8.8	7.6
	AdvPC (Ours)	98.0	**17.2**	**28.0**	**22.0**	98.8	**16.0**	**19.6**	**15.6**
PN++ (MSG)	3D-Adv [13]	6.8	100	32.4	14.8	7.6	100	28.0	14.0
	KNN [10]	7.2	100	22.8	8.4	6.8	100	22.8	8.4
	AdvPC (Ours)	**13.2**	94.8	**53.2**	**33.2**	**22.8**	98.4	**55.2**	**44.0**
PN++ (SSG)	3D-Adv [13]	6.8	8.8	100	8.0	7.2	10.4	100	7.2
	KNN [10]	6.8	8.8	100	7.6	6.4	8.4	100	6.4
	AdvPC (Ours)	**10.8**	**27.6**	96.4	**26.8**	**10.0**	**25.6**	98.8	**23.6**
DGCNN	3D-Adv [13]	10.8	14.4	39.6	100	10.8	14.0	32.4	100
	KNN [10]	7.2	11.2	13.6	100	6.8	8.4	11.2	99.6
	AdvPC (Ours)	**20.8**	**32.4**	**52.4**	85.2	**38.8**	**48.4**	**63.2**	98.8

Table 2: **Transferability of Attacks under ℓ_2 Norms**: We use norm-budgets (max ℓ_2 norm allowed in the perturbation) of $\epsilon_2 = 1.8$ and $\epsilon_2 = 4.0$. 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 [11]. 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.

5.2 Transferability on Different Norms

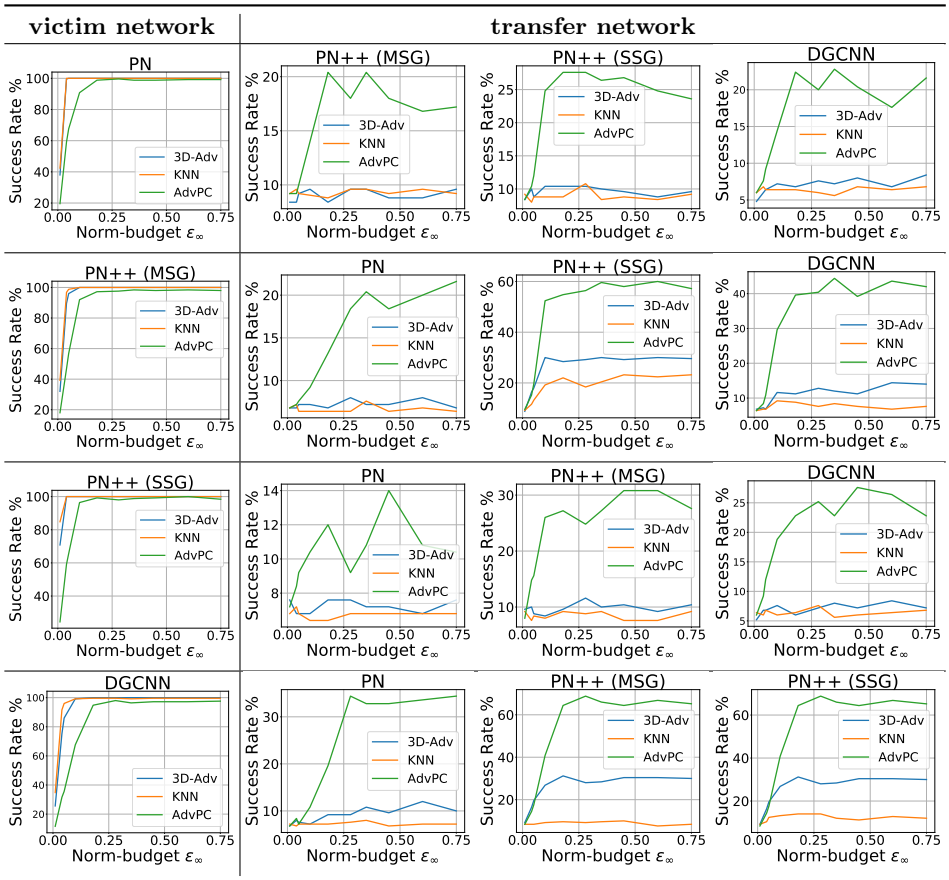

Fig. 4: **Transferability Across Different ϵ_∞ Norm-Budgets**: Here, the attacks are optimized using different ϵ_∞ norm-budgets. We report the attack success on all victim networks and the success of these attacks on each transfer network. We note that our AdvPC transfers better to the other networks across different ϵ_∞ as compared to the baselines 3D-adv[13] and KNN attack [10].

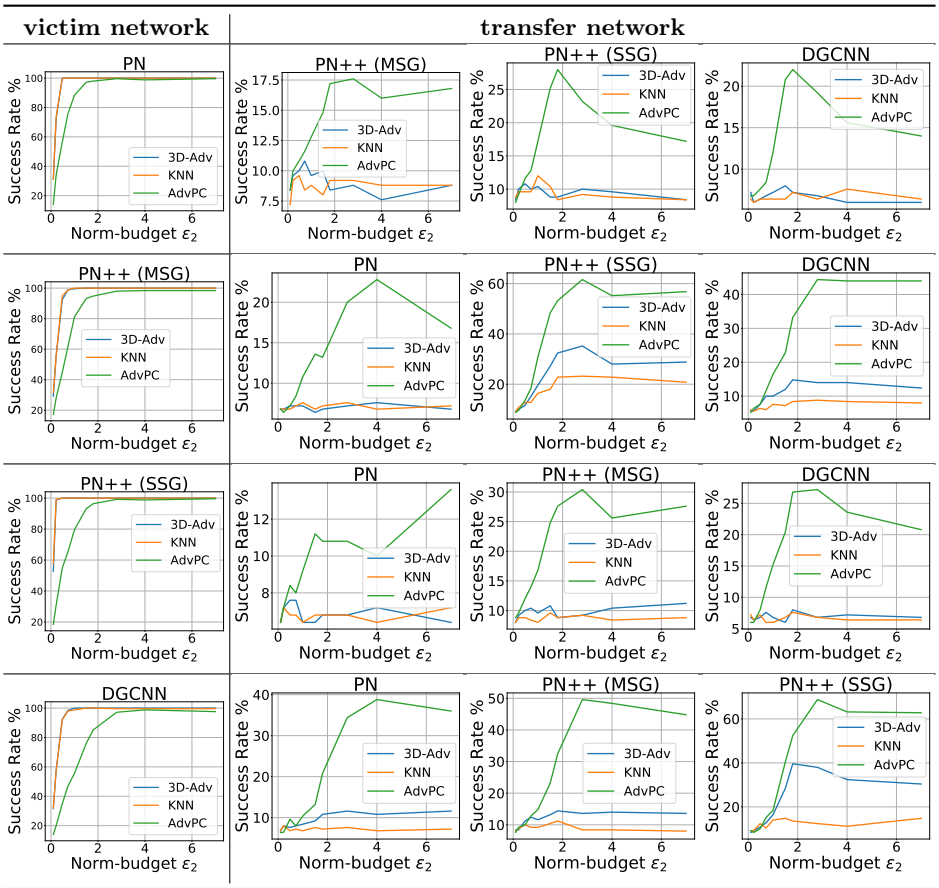

Fig. 5: **Transferability Across Different ϵ_2 Norm-Budgets**: Here, the attacks are optimized using different ϵ_2 norm-budgets. We report the attack success on all victim networks and the success of these attacks on each transfer network. We note that our AdvPC transfers better to the other networks across different ϵ_2 as compared to the baselines 3D-adv[13] and KNN attack [10].

5.3 Transferability Matrices

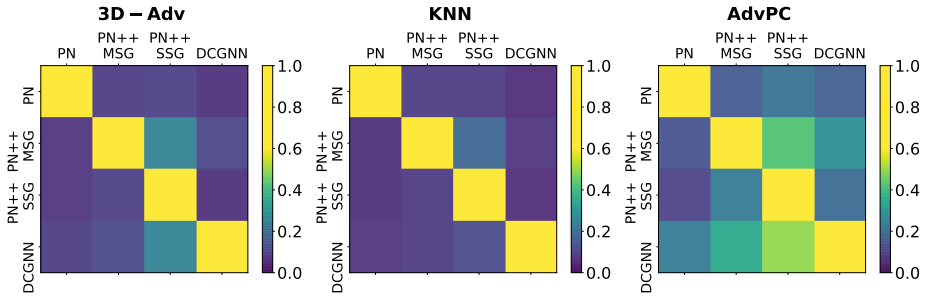

Fig. 6: **Transferability Matrix for** ℓ_∞: Visualizing the overall transferability for 3D-adv [13] (*left*), KNN attack [10](*middle*), and our AdvPC (*right*). Elements in the same row correspond to the same victim network used in the attack, while those in the same column correspond to the network 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 scores overall transferability for an attack.

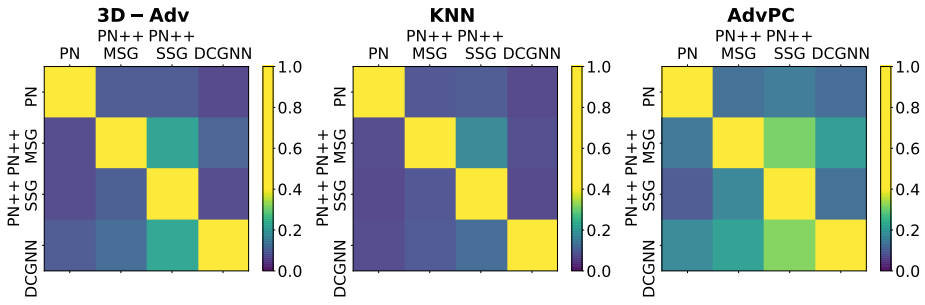

Fig. 7: **Transferability Matrix for** ℓ_2: Visualizing the overall transferability for 3D-adv [13] (*left*), KNN attack [10](*middle*), and our AdvPC (*right*). Elements in the same row correspond to the same victim network used in the attack, while those in the same column correspond to the network that the attack is transferred to. Each matrix element measures the average success rate over the range of ϵ_2 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 scores overall transferability for an attack.

6 Defenses Results (Untargeted Attacks)

6.1 ℓ_∞ Defense Results

Defenses	$\epsilon_\infty = 0.18$			$\epsilon_\infty = 0.45$		
	3D-Adv [13]	**KNN [10]**	**AdvPC (ours)**	**3D-Adv [13]**	**KNN [10]**	**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 [13]	7.2	7.6	**39.6**	8.8	7.2	**42.4**
SOR [15]	18.8	17.2	**36.8**	19.2	19.2	**32.0**
DUP Net [15]	28	28.8	**43.6**	28	31.2	**37.2**
SRS [15]	43.2	29.2	**80.0**	47.6	31.2	**85.6**

Table 3: **Attacking Point Cloud Defenses (ℓ_∞ Untargeted DGCNN):** We evaluate untargeted attacks using norm-budgets of $\epsilon_\infty = 0.18$ and $\epsilon_\infty = 0.45$ with DGCNN [11] 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 [13,10] 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).

Defenses	$\epsilon_\infty = 0.18$			$\epsilon_\infty = 0.45$		
	3D-Adv [13]	**KNN [10]**	**AdvPC (ours)**	**3D-Adv [13]**	**KNN [10]**	**AdvPC (ours)**
No defense	**100**	**100**	99.2	**100**	**100**	99.2
AE (newly trained)	14.8	13.6	**17.6**	12.0	13.2	**19.6**
Adv Training [13]	12.0	7.6	**76.4**	11.2	10.8	**76.4**
SOR [15]	20.4	18.4	**51.2**	18.8	16.0	**51.2**
DUP Net [15]	18.0	16.4	**33.6**	16.8	18.4	**38.8**
SRS [15]	53.2	40.8	**90.4**	49.2	42.4	**89.6**

Table 4: **Attacking Point Cloud Defenses (ℓ_∞ Untargeted PointNet++ SSG):** We evaluate untargeted attacks using norm-budgets of $\epsilon_\infty = 0.18$ and $\epsilon_\infty = 0.45$ with PointNet++ SSG [9] 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 [13,10] for all defenses. Note that both the attacks *and* evaluations are performed on PointNet++ SSG, which has an accuracy of 91.5% without input perturbations (for reference).

Defenses	$\epsilon_\infty = 0.18$			$\epsilon_\infty = 0.45$		
	3D-Adv [13]	**KNN [10]**	**AdvPC (ours)**	**3D-Adv [13]**	**KNN [10]**	**AdvPC (ours)**
No defense	**100**	**100**	97.2	**100**	**100**	98.0
AE (newly trained)	13.2	10.0	**20.0**	12.4	12.0	**18.4**
Adv Training [13]	6.8	26.0	**36.4**	8.0	31.2	**32.8**
SOR [15]	21.6	26.0	**53.2**	24.4	34.0	**42.4**
DUP Net [15]	29.6	27.6	**43.2**	24.8	30.8	**42.0**
SRS [15]	43.6	45.6	**80.4**	41.2	50.0	**78.8**

Table 5: **Attacking Point Cloud Defenses (ℓ_∞ Untargeted PointNet++ MSG):** We evaluate untargeted attacks using norm-budgets of $\epsilon_\infty = 0.18$ and $\epsilon_\infty = 0.45$ with PointNet++ MSG [9] 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 [13,10] for all defenses. Note that both the attacks *and* evaluations are performed on PointNet++ MSG, which has an accuracy of 91.5% without input perturbations (for reference).

Defenses	$\epsilon_\infty = 0.18$			$\epsilon_\infty = 0.45$		
	3D-Adv [13]	**KNN [10]**	**AdvPC (ours)**	**3D-Adv [13]**	**KNN [10]**	**AdvPC (ours)**
No defense	**100**	**100**	98.8	**100**	**100**	98.8
AE (newly trained)	8.0	7.6	**11.6**	8.0	7.6	**12.4**
Adv Training [13]	8.0	8.4	**41.6**	8.4	9.2	**44.8**
SOR [15]	16.0	15.6	**29.2**	16.8	15.2	**28.4**
DUP Net [15]	10.0	10.4	**12.4**	11.2	8.4	**11.2**
SRS [15]	80.8	81.6	**97.6**	85.6	77.6	**97.2**

Table 6: **Attacking Point Cloud Defenses (ℓ_∞ Untargeted PointNet):** We evaluate untargeted attacks using norm-budgets of $\epsilon_\infty = 0.18$ and $\epsilon_\infty = 0.45$ with PointNet [8] 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 [13,10] for all defenses. Note that both the attacks *and* evaluations are performed on PointNet, which has an accuracy of 92.8% without input perturbations (for reference).

6.2 ℓ_2 Defense Results

Defenses	$\epsilon_2 = 1.8$			$\epsilon_2 = 4.0$		
	3D-Adv [13]	**KNN [10]**	**AdvPC (ours)**	**3D-Adv [13]**	**KNN [10]**	**AdvPC (ours)**
No defense	**100**	**100**	85.2	**100**	99.6	98.8
AE (newly trained)	9.6	9.6	**11.6**	10.8	10.0	**21.6**
Adv Training [13]	16.8	37.6	**48.0**	8.0	13.2	**40.8**
SOR [15]	22.0	29.2	**36.8**	18.0	20.4	**27.2**
DUP Net [15]	34.8	36.0	**36.8**	28.8	28.4	**31.2**
SRS [15]	63.6	61.6	**76.0**	50.8	34.0	**88.4**

Table 7: **Attacking Point Cloud Defenses (ℓ_2 Untargeted DGCNN):** We evaluate untargeted attacks using norm-budgets of $\epsilon_2 = 1.8$ and $\epsilon_2 = 4.0$ with DGCNN [11] 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 [13,10] 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).

Defenses	$\epsilon_2 = 1.8$			$\epsilon_2 = 4.0$		
	3D-Adv [13]	**KNN [10]**	**AdvPC (ours)**	**3D-Adv [13]**	**KNN [10]**	**AdvPC (ours)**
No defense	**100**	**100**	96.4	**100**	**100**	98.8
AE (newly trained)	13.2	14.0	**18.0**	13.6	14.0	**17.6**
Adv Training [13]	20.8	19.2	**74.4**	10.8	11.6	**71.2**
SOR [15]	24.8	17.2	**49.6**	17.6	14.4	**48.4**
DUP Net [15]	18.4	15.2	**33.6**	18.0	16.0	**32.8**
SRS [15]	60.4	55.2	**86.4**	50.8	42.4	**89.2**

Table 8: **Attacking Point Cloud Defenses (ℓ_2 Untargeted PointNet++ SSG):** We evaluate untargeted attacks using norm-budgets of $\epsilon_2 = 1.8$ and $\epsilon_2 = 4.0$ with PointNet++ SSG [9] 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 [13,10] for all defenses. Note that both the attacks *and* evaluations are performed on PointNet++ SSG, which has an accuracy of 91.5% without input perturbations (for reference).

Defenses	$\epsilon_2 = 1.8$			$\epsilon_2 = 4.0$		
	3D-Adv [13]	**KNN [10]**	**AdvPC (ours)**	**3D-Adv [13]**	**KNN [10]**	**AdvPC (ours)**
No defense	**100**	**100**	94.8	**100**	**100**	98.4
AE (newly trained)	13.2	11.2	**18.4**	14.8	9.6	**20.0**
Adv Training [13]	18.8	46.0	**48.4**	8.0	34.4	**36.8**
SOR [15]	32.8	37.2	**49.2**	19.2	37.2	**47.2**
DUP Net [15]	31.6	33.6	**42.8**	26.8	32.8	**40.4**
SRS [15]	63.6	64.8	**83.6**	44.8	49.6	**80.0**

Table 9: **Attacking Point Cloud Defenses (ℓ_2 Untargeted PointNet++ MSG):** We evaluate untargeted attacks using norm-budgets of $\epsilon_2 = 1.8$ and $\epsilon_2 = 4.0$ with PointNet++ MSG [9] 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 [13,10] for all defenses. Note that both the attacks *and* evaluations are performed on PointNet++ MSG, which has an accuracy of 91.5% without input perturbations (for reference).

Defenses	$\epsilon_2 = 1.8$			$\epsilon_2 = 4.0$		
	3D-Adv [13]	**KNN [10]**	**AdvPC (ours)**	**3D-Adv [13]**	**KNN [10]**	**AdvPC (ours)**
No defense	**100**	**100**	98.0	**100**	**100**	98.8
AE (newly trained)	7.6	7.6	**13.2**	8.0	7.6	**12.8**
Adv Training [13]	9.2	10.0	**43.6**	8.8	8.4	**44.0**
SOR [15]	20.0	14.4	**27.6**	16.4	15.2	**25.6**
DUP Net [15]	12.0	9.2	**15.6**	10.4	9.2	**11.6**
SRS [15]	88.8	84.0	**96.4**	86.8	84.4	**98.4**

Table 10: **Attacking Point Cloud Defenses (ℓ_2 Untargeted PointNet):** We evaluate untargeted attacks using norm-budgets of $\epsilon_2 = 1.8$ and $\epsilon_2 = 4.0$ with PointNet [8] 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 [13,10] for all defenses. Note that both the attacks *and* evaluations are performed on PointNet, which has an accuracy of 92.8% without input perturbations (for reference).

7 Analysis of the results

We perform several analytical experiments to explore further the results obtained in so far. We perform several analytical experiments to further explore the results obtained in Sections 5.1,5.2,5.3 and 6. 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.

7.1 Ablation Study (hyperparameter γ)

Here, we study the effect of γ used in Eq (16) 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 our results (averaged over all ϵ_∞ norm-budgets) in Fig. 8 and in Fig. 9 (averaged over all ϵ_2 norm-budgets) for the four victim networks. One observation is that, while adding the AE loss with $\gamma > 0$ indeed improves transferability, it tends to deteriorate the success rate. We pick $\gamma = 0.25$ in our experiments to balance success and transferability.

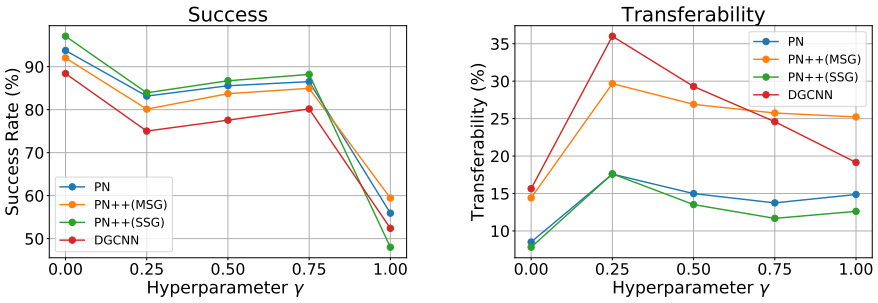

Fig. 8: **Ablation Study in ℓ_∞**: Studying the effect of changing AdvPC hyperparameter (γ) on the success rate of the attack (*left*) and on its transferability (*right*). The transferability score reported for each victim network is the average success rate on the transfer networks averaged across 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.

7.2 Network Sensitivity to Point Cloud Attacks

Fig. 10 and Fig. 11 plot the sensitivity of the various networks when they are subject to input perturbations of varying norm-budgets ϵ_∞ and ϵ_2 respectively. We measure the classification accuracy of each network under our AdvPC attack ($\gamma = 0.25$), 3D-Adv [13], and KNN attack [10]. We observe that DGCNN [11]

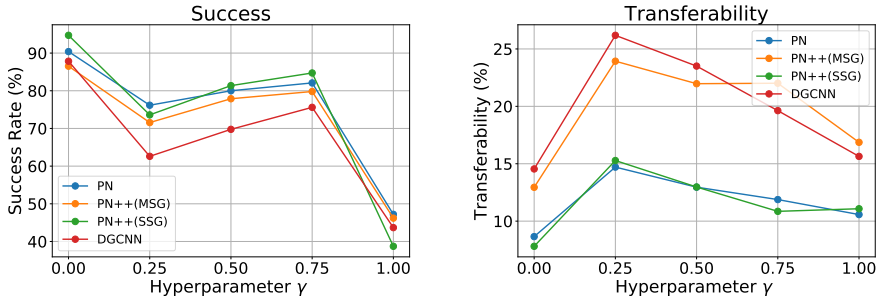

Fig. 9: **Ablation Study in ℓ_2**: Studying the effect of changing AdvPC hyperparameter (γ) on the success rate of the attack (*left*) and on its transferability (*right*). The transferability score reported for each victim network is the average success rate on the transfer networks averaged across all different norm-budgets ϵ_2. 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.

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 [11].

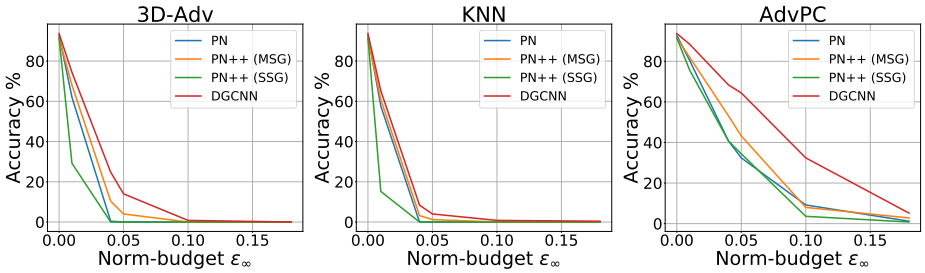

Fig. 10: **Sensitivity of Architectures in ℓ_∞**: We evaluate the sensitivity of each of the four networks for increasing norm-budget. For each network, we plot the classification accuracy under 3D-Adv perturbation [13] (*left*), KNN attack [10] (*middle*), and our AdvPC attack (*right*). Overall, DGCNN [11] is affected the least by adversarial perturbation.

7.3 Effect of the Auto-Encoder (AE)

In Fig. 12, we show an example of how AE reconstruction preserves the details of the unperturbed point cloud and does not change the classifier prediction. When

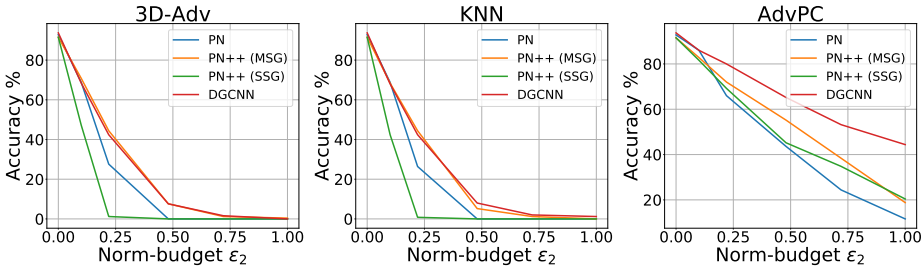

Fig. 11: **Sensitivity of Architectures in ℓ_2**: We evaluate the sensitivity of each of the four networks for increasing norm-budget. For each network, we plot the classification accuracy under 3D-Adv perturbation [13] (*left*), KNN attack [10] (*middle*), and our AdvPC attack (*right*). Overall, DGCNN [11] is affected the least by adversarial perturbation.

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 Section 6. Another observation from Fig. 12 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 (16)), 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 [13,10], which do not consider the data distribution and optimize solely for the network.

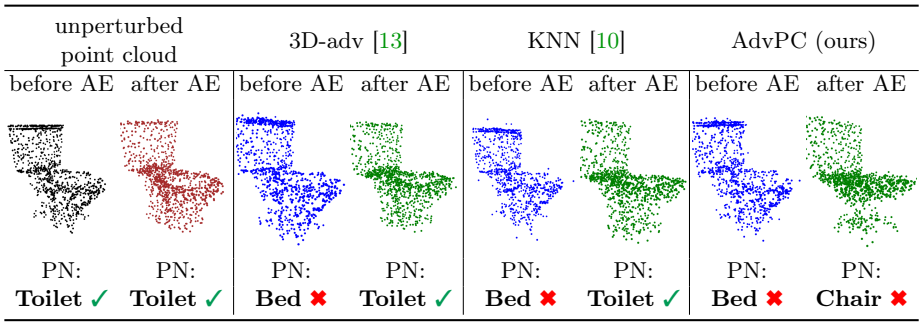

Fig. 12: **Effect of the Auto-Encoder (AE):** The AE does not affect the unperturbed point cloud (classified correctly by PN before and after AE). The AE cleans the perturbed point cloud by 3D-Adv and KNN [13,10], which allows PN to predict the correct class label. However, our AdvPC attack can fool PN before and after AE reconstruction. Perturbed samples by AdvPC, if passed through the AE, transform into similar looking objects but from different classes (Chair looks similar to Toilet).

7.4 Ablation Study on the Losses

We ablate each component of our pipeline and show their effect in our attacks. We evaluate this components by looking Attack Success Rate (ASR), transferability, and the final norm obtained under the attack. In theses experiments, we allow unconstrained attacks as well as constrained attacks. We show the effect of optimizing using EMD, CD, ℓ_2, and ℓ_∞. We show the results on Tables 11,12,13,14 for all the four networks. We observe transferability is better when using hard constraints. Constraining the attack norm allows the optimization to learn more from the AE data distribution. The EMD doesn't work well while the Chamfer loss is comparable to the ℓ_2 loss.

Attack Setup						**Results**					
soft CD	**soft EMD**	**soft ℓ_2**	**hard ℓ_∞**	**hard ℓ_2**	**AE**	**CD**	**EMD**	**ℓ_∞**	**ℓ_2**	**ASR**	**TR**
✓	-	-	-	-	-	**0.15**	4.25	0.12	**0.31**	**100**	9.02
✓	-	-	-	-	✓	0.19	5.01	0.13	0.36	99.69	9.51
-	✓	-	-	-	-	0.17	2.83	0.23	0.39	68.36	9.01
-	✓	-	-	-	✓	0.16	**2.53**	0.25	0.37	18.04	7.35
-	-	✓	-	-	-	0.16	4.38	0.11	**0.31**	**100**	8.92
-	-	✓	-	-	✓	0.21	5.22	0.13	0.36	**100**	9.35
-	-	-	✓	-	-	0.49	12.37	**0.04**	0.55	**100**	9.16
-	-	-	✓	-	✓	0.73	13.66	0.07	0.72	96.93	**13.14**
-	-	-	-	✓	-	0.26	7.41	0.09	0.38	**100**	8.87
-	-	-	-	✓	✓	0.37	7.35	0.16	0.48	99.87	11.08

Table 11: **Soft vs Hard on PointNet**: study the effect of every bit of the loss on the norms , Attack Success Rate (ASR) and Transferability (TR) under unconstrained setup vs constrained setup in PointNet [8].($\epsilon_\infty = 0.1$, $\epsilon_2 = 1.8$),$\lambda = 1$,$\gamma = 0.5$. Please refer to Section 2 for details. **Bold** numbers are the best.

Attack Setup						Results					
soft CD	**soft EMD**	**soft ℓ_2**	**hard ℓ_∞**	**hard ℓ_2**	**AE**	**CD**	**EMD**	**ℓ_∞**	**ℓ_2**	**ASR**	**TR**
✓	-	-	-	-	-	1.01	25.80	0.19	1.00	99.78	11.57
✓	-	-	-	-	✓	0.88	26.25	0.21	1.15	95.69	12.19
-	✓	-	-	-	-	0.21	5.04	0.23	0.56	14.58	6.83
-	✓	-	-	-	✓	**0.07**	**2.23**	0.12	**0.20**	2.31	6.61
-	-	✓	-	-	-	1.35	26.58	0.20	0.96	99.96	13.38
-	-	✓	-	-	✓	1.43	26.42	0.22	0.98	**100**	16.61
-	-	-	✓	-	-	3.71	53.83	**0.06**	1.84	94.71	18.46
-	-	-	✓	-	✓	2.53	38.78	0.10	1.45	97.64	**25.82**
-	-	-	-	✓	-	0.59	15.95	0.10	0.58	**100**	8.84
-	-	-	-	✓	✓	0.93	20.08	0.15	0.75	99.20	11.91

Table 12: **Soft vs Hard on PointNet++ MSG**: study the effect of every bit of the loss on the norms , Attack Success Rate (ASR) and Transferability (TR) under unconstrained setup vs constrained setup in PointNet++ MSG [9]. ($\epsilon_\infty = 0.18$, $\epsilon_2 = 1.8$),$\lambda = 1$,$\gamma = 0.5$. Please refer to Section 2 for details. **Bold** numbers are the best.

Attack Setup						Results					
soft CD	**soft EMD**	**soft ℓ_2**	**hard ℓ_∞**	**hard ℓ_2**	**AE**	**CD**	**EMD**	**ℓ_∞**	**ℓ_2**	**ASR**	**TR**
✓	-	-	-	-	-	0.25	9.39	0.07	0.37	**100**	7.01
✓	-	-	-	-	✓	0.25	9.38	0.08	0.39	99.51	7.17
-	✓	-	-	-	-	0.08	3.71	0.09	0.28	37.20	6.74
-	✓	-	-	-	✓	**0.07**	**2.95**	0.10	**0.24**	4.84	6.52
-	-	✓	-	-	-	0.30	9.90	0.07	0.39	**100**	6.92
-	-	✓	-	-	✓	0.28	9.63	0.07	0.38	**100**	7.56
-	-	-	✓	-	-	1.20	24.52	0.02	0.83	96.80	**7.84**
-	-	-	✓	-	✓	0.80	17.24	0.05	0.70	**100**	7.72
-	-	-	-	✓	-	0.19	8.08	**0.04**	0.30	**100**	6.99
-	-	-	-	✓	✓	0.46	12.42	0.09	0.50	**100**	7.44

Table 13: **Soft vs Hard on PointNet++ SSG**: study the effect of every bit of the loss on the norms , Attack Success Rate (ASR) and Transferability (TR) under unconstrained setup vs constrained setup in PointNet++ SSG [9].($\epsilon_\infty = 0.1$, $\epsilon_2 = 1.8$),$\lambda = 1$,$\gamma = 0.5$. Please refer to Section 2 for details. **Bold** numbers are the best.

Attack Setup						Results					
soft CD	**soft EMD**	**soft ℓ_2**	**hard ℓ_∞**	**hard ℓ_2**	**AE**	**CD**	**EMD**	**ℓ_∞**	**ℓ_2**	**ASR**	**TR**
✓	-	-	-	-	-	1.06	32.22	0.20	1.55	67.91	10.46
✓	-	-	-	-	✓	0.71	25.03	0.17	1.18	41.07	9.21
-	✓	-	-	-	-	0.03	2.47	0.07	0.14	2.07	7.18
-	✓	-	-	-	✓	**0.01**	**1.62**	**0.01**	**0.05**	0.76	7.21
-	-	✓	-	-	-	2.81	39.95	0.28	1.53	**99.20**	23.23
-	-	✓	-	-	✓	2.89	40.55	0.31	1.58	96.89	29.91
-	-	-	✓	-	-	4.39	53.49	0.12	2.12	86.67	26.22
-	-	-	✓	-	✓	5.10	58.24	0.16	2.40	83.56	**35.59**
-	-	-	-	✓	-	2.46	39.85	0.23	1.45	99.82	23.45
-	-	-	-	✓	✓	2.82	43.19	0.30	1.63	98.80	33.26

Table 14: **Soft vs Hard on DGCNN**: study the effect of every bit of the loss on the norms , Attack Success Rate (ASR) and Transferability (TR) under unconstrained setup vs constrained setup in DGCNN [11]. ($\epsilon_\infty = 0.18$, $\epsilon_2 = 2.8$),$\lambda = 1$,$\gamma = 0.5$. Please refer to Section 2 for details. **Bold** numbers are the best.

8 Defenses Results (Targeted Attacks)

We note from targeted attack results in Tables 15,16,17,18 that our AdvPC still outperforms the other baselines in most defenses but fail in some defenses. This can be explained because the targeted attacks with specific target label t' in Eq (15) is too strict given that the reconstruction of the AE needs to fool the classifier to unspecified label t'' that might be different from t'. This restriction makes the optimization in Eq (16) very difficult to optimize and hence leads to less successful attacks.

Defenses	$\epsilon_\infty = 0.18$			$\epsilon_\infty = 0.45$		
	3D-Adv [13]	**KNN [10]**	**AdvPC (ours)**	**3D-Adv [13]**	**KNN [10]**	**AdvPC (ours)**
No defense	77.1	**77.6**	66.4	**85.4**	80.1	77.7
AE (newly trained)	13.1	9.8	**16.5**	13.9	9.8	**18.0**
Adv Training [13]	5.1	2.5	**6.2**	5.2	2.2	**5.3**
SOR [15]	24.1	**25.6**	21.9	21.2	**26.8**	19.2
DUP Net [15]	**32.0**	30.3	27.2	30.3	**36.5**	26.7
SRS [15]	34.8	**36.2**	**36.2**	31.8	**38.7**	30.4

Table 15: **Attacking Point Cloud Defenses (ℓ_∞ Targeted DGCNN):** We evaluate targeted attacks using norm-budgets of $\epsilon_\infty = 0.18$ and $\epsilon_\infty = 0.45$ with DGCNN [11] as the victim network under different defenses for 3D point clouds. Similar to before, we report 1 - accuracy (**higher** indicates better attack). AdvPC consistently outperforms the other attacks [13,10] 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).

Defenses	$\epsilon_\infty = 0.18$			$\epsilon_\infty = 0.45$		
	3D-Adv [13]	**KNN [10]**	**AdvPC (ours)**	**3D-Adv [13]**	**KNN [10]**	**AdvPC (ours)**
No defense	99.5	**100**	98.3	99.8	**100**	98.5
AE (newly trained)	13.0	12.9	**15.5**	12.6	12.8	**14.7**
Adv Training [13]	9.3	10.3	**25.2**	5.3	10.7	**14.6**
SOR [15]	16.9	16.6	**18.0**	13.5	**20.4**	15.2
DUP Net [15]	17.3	17.4	**18.5**	15.8	**18.7**	16.9
SRS [15]	17.3	17.4	**60.8**	35.4	51.4	**53.0**

Table 16: **Attacking Point Cloud Defenses (ℓ_∞ Targeted PointNet++ SSG):** We evaluate targeted attacks using norm-budgets of $\epsilon_\infty = 0.18$ and $\epsilon_\infty = 0.45$ with PointNet++ SSG [9] as the victim network under different defenses for 3D point clouds. Similar to before, we report 1 - accuracy (**higher** indicates better attack). AdvPC consistently outperforms the other attacks [13,10] for all defenses. Note that both the attacks *and* evaluations are performed on PointNet++ SSG, which has an accuracy of 91.5% without input perturbations (for reference).

Defenses	$\epsilon_\infty = 0.18$			$\epsilon_\infty = 0.45$		
	3D-Adv [13]	**KNN [10]**	**AdvPC (ours)**	**3D-Adv [13]**	**KNN [10]**	**AdvPC (ours)**
No defense	96.6	**99.9**	94.7	99.2	**99.9**	97.6
AE (newly trained)	13.6	12.8	**16.7**	16.1	12.0	**23.2**
Adv Training [13]	7.2	3.6	**11.8**	6.6	3.6	**12.7**
SOR [15]	**32.1**	27.1	31.6	23.8	**31.3**	25.0
DUP Net [15]	**36.6**	9.6	36.2	27.6	**31.3**	30.6
SRS [15]	46.1	44.9	**57.0**	50.3	41.4	**60.3**

Table 17: **Attacking Point Cloud Defenses (ℓ_∞ Targeted PointNet++ MSG):** We evaluate targeted attacks using norm-budgets of $\epsilon_\infty = 0.18$ and $\epsilon_\infty = 0.45$ with PointNet++ MSG [9] as the victim network under different defenses for 3D point clouds. Similar to before, we report 1 - accuracy (**higher** indicates better attack). AdvPC consistently outperforms the other attacks [13,10] for all defenses. Note that both the attacks *and* evaluations are performed on PointNet++ MSG, which has an accuracy of 91.5% without input perturbations (for reference).

Defenses	$\epsilon_\infty = 0.18$			$\epsilon_\infty = 0.45$		
	3D-Adv [13]	**KNN [10]**	**AdvPC (ours)**	**3D-Adv [13]**	**KNN [10]**	**AdvPC (ours)**
No defense	**100**	**100**	97.4	**100**	**100**	98.4
AE (newly trained)	9.9	9.4	**12.6**	8.5	0.0	**9.7**
Adv Training [13]	12.2	14.6	**22.0**	11.3	**28.0**	13.2
SOR [15]	**11.2**	10.9	10.7	**9.6**	4.0	8.6
DUP Net [15]	8.5	**9.7**	7.8	8.0	**9.6**	7.8
SRS [15]	70.7	63.8	**81.4**	52.0	52.0	**63.7**

Table 18: **Attacking Point Cloud Defenses (ℓ_∞ Targeted PointNet):** We evaluate targeted attacks using norm-budgets of $\epsilon_\infty = 0.18$ and $\epsilon_\infty = 0.45$ with PointNet [8] as the victim network under different defenses for 3D point clouds. Similar to before, we report 1 - accuracy (**higher** indicates better attack). AdvPC consistently outperforms the other attacks [13,10] for all defenses. Note that both the attacks *and* evaluations are performed on PointNet, which has an accuracy of 92.8% without input perturbations (for reference).

9 Other Tried Approaches (Less Successful)

9.1 Point Cloud GAN:

We try to use the l-GAN and r-GAN from [1] to create more natural attacks to the input point clouds. We try to leverage the discriminator signal of both r-GAN and l-GAN to differentiate between the perturbed point clouds and the original samples. The idea is that if the trained discriminator can distinguish between the unperturbed and attacked samples, then we add the discriminator loss as an additional loss to the attack objective in Eq (11) to craft a perturbation that passes the trained discriminator test of natural input. We train l-GAN and r-GAN with the same procedure advised by [1] and on the same data as our AE $\mathbb{G}$. However, as Fig. 13 illustrates, neither l-GAN nor-GAN were able to distinguish between the unperturbed samples and the perturbed samples using the attack from [13]. This disappointing result leads us to abandon the approach in favor of the AE optimization (which works).

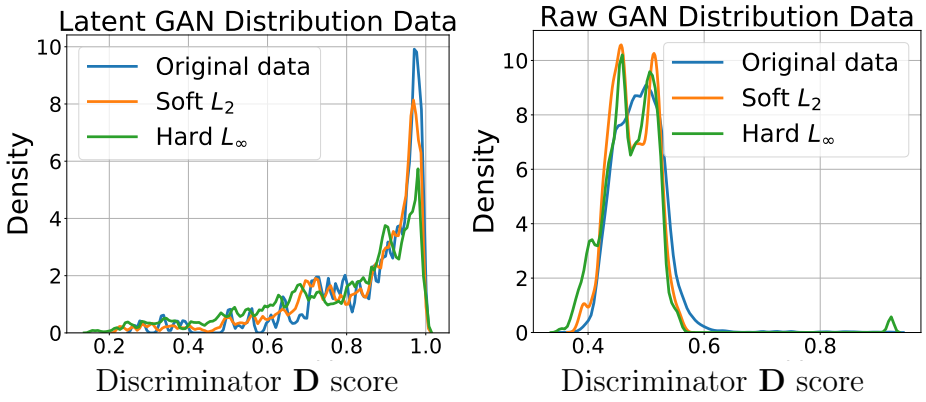

Fig. 13: **GAN instead of AE:** We tried to use l-GAN and r-GAN from [1] as natural priors for AdvPC attacks instead of the AE. The discriminators of Both l-GAN and r-GAN could not discriminate between the original data and the attacks data by soft ℓ_2 loss or Hard ℓ_∞. We show the histogram distribution of discriminator scores of the original data and attacked data using l-GAN discriminator (*left*) and r-GAN discriminator (*right*).

9.2 Learning Approach

Inspired by the success of [6,7] in learning to attack, we tried to learn the AE **G** that produces the desired perturbed point cloud $\mathcal{X}'$ by optimizing the output of the AE by the soft adversarial loss. To achieve this, the AE should output points clouds that are close as possible to the input point cloud $\mathcal{X}'$ (by the

Chamfer soft loss as in Eq (9)) and also the output of the AE should fool the classifier **F**. We note that because of the nature of point cloud, we could not project the output of the AE back to the original sample with some norm, as performed by [6], and hence we used the soft Chamfer loss instead. We train the AE to perform untargeted and targeted attacks on the training set of ModelNet40 [12] and evaluate the adversary **G** on the test set of ModelNet. We report the results of targeted and untargeted attacks in Table 19. We note that for the untargeted attacks that indeed succeed, the final Chamfer Distance is way bigger than the ones obtained by optimization (see Tables 11). This might be attributed to the difficulty of learning an attack that works under varying distance penalties, unlike the [6] where the adversarial objective is hardly conditioned on a constant distance between the attacked image and the original image.

Loss	λ_{CD}	**Learning rate**	**Untargeted/ Targeted**	**Training Epochs**	**Accuracy**	**Chamfer Distance**
Relativistic	0	0.0001	Untargeted	15	6.375	2.6161
Relativistic	1	0.0001	Untargeted	15	**5.0833**	2.4832
Relativistic	3	0.0001	Untargeted	13	9.9167	0.024229
Relativistic	10	0.0001	Untargeted	13	8.9583	0.022948
Relativistic	30	0.0001	Untargeted	13	10.125	0.021558
Relativistic	100	0.0001	Untargeted	13	13.625	0.018275
Relativistic	300	0.0001	Untargeted	16	17.875	0.014084
Relativistic	1000	0.0001	Untargeted	13	38.0417	0.09286
Relativistic	0	0.00005	Untargeted	13	9.7083	0.023431
Relativistic	0	0.0001	Targeted	0	81.5	**0.005556**
Relativistic	0	0.001	Targeted	19	31.875	0.015868

Table 19: **The Learning Approach on PointNet:** We tried to learn a network to attack PointNet [8] (approach similar to [6] but on point clouds). While the approach mildly succeeds on untargeted attacks, the final average Chamfer distance on the succeeding attacks are much bigger than those obtained by optimization like in Fig. 11. This implies that the optimization is actually better on point lcouds.

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. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: IEEE Symposium on Security and Privacy (SP) (2017)
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
4. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial machine learning at scale. CoRR **abs/1611.01236** (2016)
5. 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)
6. 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)
7. 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)
8. 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)
9. 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)
10. Tsai, T., Yang, K., Ho, T.Y., Jin, Y.: Robust adversarial objects against deep learning models. In: AAAI Conference on Artificial Intelligence (2020)
11. 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)
12. 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)
13. 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)
14. Zheng, T., Chen, C., Yuan, J., Li, B., Ren, K.: Pointcloud saliency maps. In: The IEEE International Conference on Computer Vision (ICCV) (2019)
15. 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)