Robust Structured Declarative Classifiers for 3D Point Clouds: Defending Adversarial Attacks with Implicit Gradients

Kaidong Li[1], Ziming Zhang[2]*, Cuncong Zhong[1], Guanghui Wang[3]
[1]Department of EECS, University of Kansas, KS, USA
[2]Department of ECE, Worcester Polytechnic Institute, MA, USA
[3]Department of CS, Ryerson University, Toronto ON, Canada
{kaidong.li, cczhong}@ku.edu, zzhang15@wpi.edu, wangcs@ryerson.ca

Abstract

Deep neural networks for 3D point cloud classification, such as PointNet, have been demonstrated to be vulnerable to adversarial attacks. Current adversarial defenders often learn to denoise the (attacked) point clouds by reconstruction, and then feed them to the classifiers as input. In contrast to the literature, we propose a family of robust structured declarative classifiers for point cloud classification, where the internal constrained optimization mechanism can effectively defend adversarial attacks through implicit gradients. Such classifiers can be formulated using a bilevel optimization framework. We further propose an effective and efficient instantiation of our approach, namely, Lattice Point Classifier (LPC)*, based on structured sparse coding in the permutohedral lattice and 2D convolutional neural networks (CNNs) that is end-to-end trainable. We demonstrate state-of-the-art robust point cloud classification performance on ModelNet40 and ScanNet under seven different attackers. For instance, we achieve* **89.51%** *and* **83.16%** *test accuracy on each dataset under the recent JGBA attacker that outperforms DUP-Net and IF-Defense with PointNet by ∼70%. The demo code is available at* https://zhang-vislab.github.io.

1. Introduction

Point clouds are unstructured data which is widely used in real-world applications such as autonomous driving. To recognize them using deep neural networks, point clouds can be represented as points [36], images [30], voxels [61], or graphs [56]. Recent works [74, 63, 31, 14, 62, 58, 28] have demonstrated that such deep networks are vulnerable to (gradient-based) adversarial attacks. Accordingly, several adversarial defenders [75, 60, 29] have been proposed for robust point cloud classification. The basic ideas are often to denoise the (attacked) point clouds before feeding them into the classifiers as input to preserve their prediction accuracy.

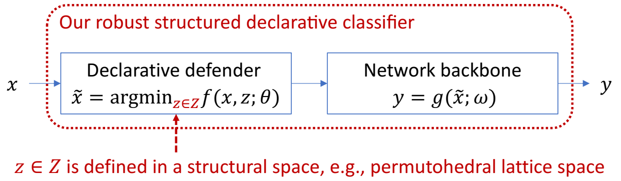

Our robust structured declarative classifier
x
Declarative defender
$\tilde{x} = \text{argmin}_{z \in Z} f(x, z; \theta)$
Network backbone
$y = g(\tilde{x}; \omega)$
y
$z \in Z$ is defined in a structural space, e.g., permutohedral lattice space

Figure 1: Illustration of robust structured declarative classifiers, where our defender is optimized in a structural space.

Obfuscated gradients. In the white-box adversarial attacks, the attackers are assumed to have full access to both classifiers and defenders. To defend such attacks, one common way is to *break* the gradient over the input data in the backpropagation (either inadvertently or intentionally, *e.g.,* a defender is non-differentiable or prevents gradient signal from flowing through the network) so that the attackers fail to be optimized. Such scenarios are called obfuscated gradients. In [4] Athalye *et al.* have discussed the false sense of security in such defenders and proposed new methods, such as Backward Pass Differentiable Approximation (BPDA), to attack them successfully. Take DUP-Net [75] for example, where a non-differentiable Statistical Outlier Removal (SOR) defense strategy was proposed. In [31] Ma *et al.* proposed Joint Gradient Based Attack (JGBA) that can compute the gradient with a linear approximation (an instantiation of BPDA) of the SOR defense to attack DUP-Net successfully.

Implicit gradients. Now let us consider the scenarios where both defenders and classifiers are differentiable. Then in order to defend the adversarial attacks, one way is to make the calculation of the gradient challenging. To this end, implicit gradients [40] may be more suitable for designing the defenders. *An implicit gradient,* $\frac{\partial y}{\partial x}$*, is defined by a differentiable function* h *that takes* x, y *as its input,* i.e., $\frac{\partial y}{\partial x} = h(x, y)$. Such an equation can be also considered as a first-order ordinary differential equation (ODE), which is

*Joint first author

solvable (approximately) using Euler's Method [19]. Here we assume that the gradients through the classifiers can be easily computed, which often holds empirically. To our best knowledge, so far there is no work on designing adversarial defenders for 3D point clouds based on implicit gradients.

Declarative networks. Implicit gradients require equations that contain both the input and output of a defender. One potential solution for this is to introduce optimization problems as the defenders, where the first-order optimality conditions provide such equations. In the literature, there have been some works [3, 2, 25, 40] that proposed optimization as network layers in deep neural networks. Recently in [12] Gould *et al.* generalized these ideas and proposed deep declarative networks. A *declarative* network node is introduced where the exact implementation of the forward processing function is not defined; rather the input-output relationship ($x \mapsto \tilde{x}$) is defined in terms of behavior specified as the solution to an optimization problem $\tilde{x} \in \arg\min_{z\in\mathcal{Z}} f(x, z; \theta)$. Here f is an objective function, θ denotes the node parameters, and $\mathcal{Z}$ is the feasible solution space. In [12] a robust pooling layer was proposed as a declarative node using unconstrained minimization with various penalty functions such as Huber or Welsch, which can be efficiently solved using Newton's method or gradient descent. The effectiveness of such pooling layers was demonstrated for point cloud classification.

Our approach. Motivated by the methods above, in this paper we propose a novel robust structured declarative classifiers for 3D point clouds by embedding a declarative node into the networks, as illustrated by Fig. 1. Different from robust pooling layers in [12], our declarative defender is designed to reconstruct each point cloud in a (learnable) structural space as a means of denoising. To this end, we borrow the idea from structured sparse coding [49, 20, 57] by representing each point as a linear combination of atoms in a dictionary. Together with the backbone networks, the training of our robust classifiers can lead to a bilevel optimization problem. Considering the inference efficiency, one plausible instantiation of our classifiers, as illustrated in Fig. 2, is to define the structural space using the permutohedral lattice [1, 47, 13], project each point cloud onto the lattice, generate a 2D image based on the barycentric weights, and feed the image to a 2D convolutional neural network (CNN) for classification. We call this instantiation *Lattice Point Classifier (LPC).*

Our contributions. We summarize our contributions below:

- We propose a family of novel robust structured declarative classifiers for 3D point clouds where the declarative nodes defend the adversarial attacks through implicit gradients. To the best of our knowledge, we are the *first* to explore implicit gradients in robust point cloud classification.
- We propose a bilevel optimization framework to learn the network parameters in an end-to-end fashion.

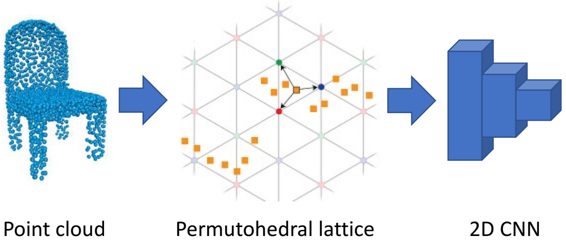


Figure 2: Illustration of our Lattice Point Classifier (LPC).

- We propose an effective and efficient instantiation of our robust classifiers based on the structured sparse coding in the permutohedral lattice and 2D CNNs.
- We demonstrate superior performance of our approach by comparing with the state-of-the-art adversarial defenders under the state-of-the-art adversarial attackers.

2. Related Works

Deep learning for 3D point clouds. Based on the point cloud representations, we simply group some typical deep networks into four categories. *Point-based networks* [36, 38, 26, 65, 59] directly take each point cloud as input, extract point-wise features using multi-layer perceptrons (MLPs), and fuse them to generate a feature for the point cloud. *Image-based networks* [48, 37, 68, 67, 30, 33] often project a 3D point cloud onto a (or multiple) 2D plane to generate a (or multiple) 2D image for further process. *Voxel-based networks* [32, 61, 41, 55, 24] usually voxelize each point cloud into a volumetric occupancy grid and further some classification techniques such as 3D CNNs are used for the tasks. *Graph-based networks* [23, 56, 44, 10, 39] often represent each point cloud as a graph such as KNN or adjacency graph which are fed to train graph convolutional networks (GCNs). A nice survey can be found in [15].

Adversarial attacks on point clouds. Such attacks aim to modify the input point clouds in a way that is not noticeable but can fool a classifier. Attackers can either have a target or not. Targeted attackers try to fool the classifier to predict a specified wrong class, while untargeted ones do not care about the predicted class as long as it is wrong. Nice surveys on adversarial attacks can be found in [34, 54, 5]. Below we summarize some typical attackers for point clouds:

- *Point perturbation [27, 66, 62, 21, 31].* Inspired by Fast Gradient Sign Method (FGSM) [11], they add on each point a small perturbation constrained by distance metrics (*e.g.,* L_p norm [62], on the surface of an ϵ-ball [27]).
- *Point addition [62, 58, 66].* Independent point attackers initially pick some points from target classes, add small perturbations and finally append these points to the victim point clouds. Cluster attackers similarly pick most critical points and then find a specified number of small point clusters to append to the victim point clouds. Object

attackers attach scaled and rotated foreign object to the original point clouds.

- *Point dropping.* Adversarial attackers pick critical points from each input point cloud, and drop them to fool the classifier. However, it is a non-differentiable operation. To address the issue in learning, [73] proposed creating saliency maps by viewing dropping a point as moving it to the cloud center. Wicker *et al.* [58] designed an algorithm to iteratively find the points to drop by minimizing a predefined objective function, similarly in [66].
- *Others.* Hamdi *et al.* [16] proposed a transferable adversarial perturbation attacker based on an adversarial loss that can learn the data distribution. Zhao *et al.* [72] proposed a black-box (*i.e.,* no access to the models) attacker with zero loss in isometry, as well as a white-box (*i.e.,* full access to the models) attacker based on the spectral norm. LG-GAN [74] is generative adversarial network (GAN) based, which learns and incorporates target features into victim point clouds. A backdoor attacker was proposed in [63] to trick the 3D models by inserting adversarial point patterns into the training set so that the victim models learn to recognize the adversarial patterns during inference.

Adversarial defense on point clouds. Adversarial defenses aim to denoise the input point clouds to recover the ground-truth labels from the classifiers. Nice surveys on adversarial defenses can be found in [34, 54, 5]. Below we summarize some typical defenders for point clouds:

- *Statistical outlier removal (SOR) [43].* SOR can be used to remove local roughness on the (smooth) surface as a means of defense. SOR is not differentiable, producing obfuscated gradients for defenders. DUP-Net [75] uses SOR and an upsampling network to reconstruct higher resolution point clouds. Similarly, IF-Defense [60] utilizes SOR, followed by a geometry-aware model to encourage evenly distributed points. However, such defenders have been demonstrated to be attackable in [52, 31]. Dong *et al.* [8] proposed replacing SOR by attention mechanism.
- *Random sampling.* Yang *et al.* [66] suggested that 3D models with random sampling are robust to adversarial attacks. PointGuard [29] proposed majority voting for point cloud classification by predicting multiple randomly subsampled point clouds.
- *Data augmentation.* Tramer *et al.* [51] demonstrated that data augmentation can effectively account for adversarial attacks. Tu *et al.* [53] proposed generating physically realizable adversarial examples to train robust Lidar object detectors. Zhang *et al.* [70] proposed randomly permuting training data as a simple data augmentation strategy. PointCutMix [69] pairs two training clouds and swaps some points between the pair to generate new training data.

Permutohedral lattice. Permutohedral lattice is a powerful operation to project the coordinates from a high dimensional space onto a hyperplane that defines the lattice. It has been widely used in high dimensional filtering [1, 18] that consists of three components, *i.e.,* splat, blur and slice. In particular, we illustrate the splat in Fig. 2, where each square represents a projection (*i.e.,* projected point) from a 3D coordinate. The splat first locates the enclosing lattice simplex for the 3D point and calculates the vertex coordinates of the simplex. Then each projection distributes its value to the vertices using barycentric interpolation with barycentric weights that are calculated as the normalized triangular areas between the projection and any pair of its corresponding lattice vertices. We refer the readers to [1] for more details. Recently permutohedral lattice has been successfully explored in point cloud segmentation [47, 13, 42] with remarkable performance.

3. Robust Structured Declarative Classifiers

3.1. Structured Declarative Defender

Recall that adversarial defenders often aim to denoise the input point clouds by reconstructing them in certain ways, and sparse coding [71] is one of the classic approaches for finding a sparse representation of the input data in the form of a linear combination of basic elements as well as those basic elements themselves. Due to its simplicity, we consider using sparse coding as a means to construct declarative nodes.

Specifically, in the 3D space given a (learnable) dictionary $\mathbf{B} \in \mathbb{R}^{3\times N}$ with $N \gg 3$ atoms, (structured) sparse coding aims to solve the following optimization problem for each point $\mathbf{x}_i \in \mathbb{R}^3$ in a point cloud $\mathbf{x} = \{\mathbf{x}_i\} \subseteq \mathbb{R}^3$:

$$\tilde{\mathbf{x}}_i \in \arg\min_{\mathbf{z}\in\mathcal{Z}} f(\mathbf{x}_i, \mathbf{z}) = \frac{1}{2}\|\mathbf{x}_i - \mathbf{B}\mathbf{z}\|^2 + \phi(\mathbf{z}), \qquad (1)$$

where ϕ denotes a regularization term, and $\mathcal{Z} \subseteq \mathbb{R}^N$ denotes a structural feasible solution space.

Obfuscated & implicit gradients in $\frac{\partial \tilde{\mathbf{x}}_i}{\partial \mathbf{x}_i} = \left.\frac{\partial \mathbf{z}}{\partial \mathbf{x}_i}\right|_{\mathbf{z}=\tilde{\mathbf{x}}_i}$. To see this, we can rewrite Eq. (1) as $\tilde{\mathbf{x}}_i \in \arg\min_{\mathbf{z}} F(\mathbf{x}_i, \mathbf{z}) = f(\mathbf{x}_i, \mathbf{z}) + \phi(\mathbf{z}) + \delta(\mathbf{z})$, where $\delta(\mathbf{z})$ denotes the Dirac delta function returning 0 if $\mathbf{z} \in \mathcal{Z}$ holds, otherwise, $+\infty$. Therefore, F will become non-differentiable when $\mathbf{z} \notin \mathcal{Z}$ or ϕ is non-differentiable over $\mathbf{z} \in \mathcal{Z}$, leading to obfuscated gradients. Otherwise, based on the first-order optimality condition, we have $\mathbf{B}^T\mathbf{B}\tilde{\mathbf{x}}_i - \mathbf{B}^T\mathbf{x}_i + \phi'(\tilde{\mathbf{x}}_i) = \mathbf{0}$, where $(\cdot)^T$ denotes the matrix transpose operator and ϕ' denotes the first-order derivative of ϕ. By taking another derivative on both sides, we then have a linear system $\left(\mathbf{B}^T\mathbf{B} + \phi''(\tilde{\mathbf{x}}_i)\right) \cdot \frac{\partial \tilde{\mathbf{x}}_i}{\partial \mathbf{x}_i} = \mathbf{B}^T$, if the second-order derivative, ϕ'', exists at $\mathbf{z} \in \mathcal{Z}$. Clearly, solving this linear system may be challenging, because $\phi''(\tilde{\mathbf{x}}_i)$ may not be computable and the matrix $\mathbf{B}^T\mathbf{B} + \phi''(\tilde{\mathbf{x}}_i)$ may be rank-deficient. Such phenomenons lead to implicit gradients.

3.2. Parameter Learning via Bilevel Optimization

Fig. 1 illustrates our approach with two components, *i.e.,* a declarative defender f and a network backbone g that takes the outputs of the defender as input and then makes predictions. Equivalently we can formulate the training of such networks as a bilevel optimization problem as follows:

$$\min_{\mathbf{B},\omega} \sum_{(\mathbf{x},y)\in\mathcal{X}\times\mathcal{Y}} \ell\Big(g(\tilde{\mathbf{x}};\omega),y\Big), \tag{2}$$

$$\text{s.t. } \tilde{\mathbf{x}}_i \in \arg\min_{\mathbf{z}\in\mathcal{Z}} \frac{1}{2}\|\mathbf{x}_i - \mathbf{B}\mathbf{z}\|^2 + \phi(\mathbf{z}), \forall \mathbf{x}_i \in \mathbf{x},$$

where $(\mathbf{x}, y) \in \mathcal{X}\times\mathcal{Y}$ denotes a training sample with data $\mathbf{x}$ and label y, ℓ denotes a loss function such as cross-entropy, and $\mathbf{B}, \omega$ denote the defender and network parameters, respectively. Same as training deep networks, we can solve this optimization problem using (stochastic) gradient descent.

Validity of $\frac{\partial g}{\partial \tilde{\mathbf{x}}_i} = \frac{\partial g}{\partial \mathbf{z}}\Big|_{\mathbf{z}=\tilde{\mathbf{x}}_i}$ in the structural space. In a gradient based adversarial attack, the gradient for modifying an input point $\mathbf{x}_i$ through backpropagation can be written as $\frac{\partial g}{\partial \tilde{\mathbf{x}}_i}\frac{\partial \tilde{\mathbf{x}}_i}{\partial \mathbf{x}_i}$. Assuming that $\frac{\partial \tilde{\mathbf{x}}_i}{\partial \mathbf{x}_i}$ can be computed exactly, then the gradient in the attack would hold in general only if $\mathbf{z}$ was unconstrained so that $\frac{\partial g}{\partial \tilde{\mathbf{x}}_i}$ is valid. Unfortunately in our case this is not true as we constrain $\mathbf{z} \in \mathcal{Z}$. Therefore, the attack in the declarative node will produce inaccurate gradients, and such errors will be propagated to the adversarial examples, leading to failure cases together with $\frac{\partial \tilde{\mathbf{x}}_i}{\partial \mathbf{x}_i}$.

3.3. Instantiation: Lattice Point Classifier

So far we have explained the learning principles and defense philosophy in our approach. The key challenge now is how to design the structural space $\mathcal{Z}$ and the regularizer ϕ to achieve robust point cloud classifiers that can be trained and inferred effectively and efficiently. To address this issue, we borrow the idea from permutohedral lattice, and propose an instantiation, namely, Lattice Point Classifier (LPC).

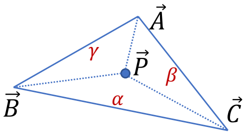


Figure 3: Illustration of barycentric coordinates and weights.

Geometric view on barycentric coordinates and their weights. Barycentric coordinates ($\vec{A}$, $\vec{B}$, $\vec{C}$ in Fig. 3) can be used to express the position of any point ($\vec{P}$) located on the entire triangle with three scalar weights (α, β, γ). To compute $\vec{P}$ using barycentric coordinates we can always use the following equation:

$$\vec{P} = \alpha\vec{A} + \beta\vec{B} + \gamma\vec{C}, \exists\alpha,\beta,\gamma \geq 0, \alpha+\beta+\gamma = 1. \tag{3}$$

Note that this equation holds for an arbitrary dimensional space, including the 3D space. Now given $\vec{A}, \vec{B}, \vec{C}, \vec{P}$, we can compute α, β, α using the normalized areas. Taking γ for example, we can compute it as follows:

$$\gamma = \frac{\|\overrightarrow{AB} \times \overrightarrow{AP}\|}{\|\overrightarrow{AB} \times \overrightarrow{AC}\|} \propto \|\overrightarrow{AB} \times \overrightarrow{AP}\|, \tag{4}$$

where $\overrightarrow{AB} = \vec{B} - \vec{A}, \overrightarrow{AP} = \vec{P} - \vec{A}, \overrightarrow{AC} = \vec{C} - \vec{A}$, $\times$ denotes the cross product operator, and $\|\cdot\|$ denotes the ℓ_2 norm of a vector measuring its length. Clearly, the barycentric weights define a nonlinear mapping that can be computed efficiently using the splat operation for the permutohedral lattice.

Constructing $\mathcal{Z}$ and ϕ using barycentric weights. By substituting Eq. (3) into Eq. (1), we manage to define a structured sparse coding problem, where the structural space $\mathcal{Z}$ and the regularizer ϕ can be constructed as follows:

$$\mathcal{Z} \overset{def}{=} \left\{\mathbf{z} \mid \mathbf{z}^T\mathbf{e} = 1, \mathbf{z} \succeq \mathbf{0}\right\}, \tag{5}$$

$$\phi(\mathbf{z}) \overset{def}{=} \lambda\sum_{n=1}^{N} \|\mathbf{B}\mathbf{z} - \mathbf{B}_n\| \cdot 1_{\{\mathbf{z}_n>0\}}, \tag{6}$$

where $\mathbf{e}$ denotes a vector of 1's, $\succeq$ denotes the entry-wise operator of $\geq$, $\mathbf{B}_n \in \mathbb{R}^3$ denotes the n-th column in $\mathbf{B}$, $1_{\{\cdot\}}$ denotes a binary indicator returning 1 if the condition holds, otherwise 0 (*i.e.,* the *binarization* of barycentric weights), and $\lambda \geq 0$ is a small constant controlling the contribution of ϕ to the objective so that it will not be dominated by ϕ.

Proposition 1. *Supposing that* $\mathbf{B}$ *in Eq.* (1) *represents the vertices in a permutohedral lattice that is large enough to cover all possible projections from points among the data, then there exists a solution to minimize the reconstruction loss using three vertices, at most, and the minimum loss is equal to the projection loss onto the lattice.*

This is because Eq. (3) defines a lossless representation using a linear combination of three vertices. The only loss occurs when projecting a point to the permutohedral lattice.

Defense mechanism in LPC. Fig. 4 illustrates how our declarative defender works with the permutohedral lattice, where the triangle, circles and squares represent a lattice cell, 3D points and their projections on the cell, respectively. In the adversarial point cloud, the attacker modifies a point from $\mathbf{x}_i$ to $\mathbf{x}'_i$. Then during the inference, our defender projects $\mathbf{x}'_i$ to $\tilde{\mathbf{x}}'_i$ on the lattice.

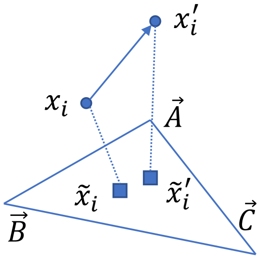


Figure 4: Illustration of our defense mechanism.

Proposition 2. *Supposing* $\tilde{\mathbf{x}}_i = \alpha\vec{A} + \beta\vec{B} + \gamma\vec{C}$ *where* α, β, γ *are the barycentric weights, then in order to guarantee that* $\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}'_i$ *lie in different cells, the distance between* $\mathbf{x}_i$ *and* $\mathbf{x}'_i$ *should be bigger than the shortest distance between* $\tilde{\mathbf{x}}'_i$ *and the boundary of the triangle, that is:*

$$\|\mathbf{x}_i - \mathbf{x}'_i\| > \min\left\{\frac{\alpha}{\|\overrightarrow{BC}\|}, \frac{\beta}{\|\overrightarrow{AC}\|}, \frac{\gamma}{\|\overrightarrow{AB}\|}\right\} \cdot s, \tag{7}$$

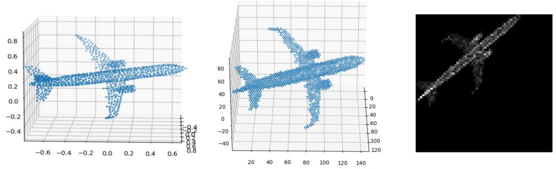


Figure 5: Illustration of **(left)** a point cloud, **(middle)** its lattice representation, **(right)** its image for classification.

where s denotes the area of the triangle. In particular, if the triangle is equilateral, then $\|\mathbf{x}_i - \mathbf{x}'_i\| > \frac{\sqrt{3}}{2} l \cdot \min\{\alpha, \beta, \gamma\}$ where l denotes the side length.

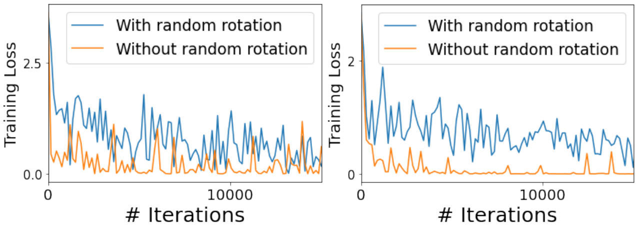
It will be more intuitive to understand this result if we binarize the barycentric weights before feeding them into the backbone network for classification, because different projections lying in the same lattice cell will lead to the same representation. This could be an effective way to remove the adversarial noise in the data. Also, this result indicates that (1) the points whose projections are closer to the boundary are easier to change their sparse representations, (2) the movements that make such changes are proportional to the scale of the lattice cell. In general, larger cells will be more tolerant to the adversarial noise, but they may sacrifice the generalization of the classifiers.

Workflow of LPC. To summarize, it is operated as follows:

1. Given a point cloud, the barycentric weights of each point are computed using the splat for permutohedral lattice;
2. Generate an image for the point cloud by averaging the barycentric weights over all the points (after binarization if applied) and aligning the lattice with the image representation (see Fig. 5 for illustration);
3. Apply a 2D CNN as the backbone network to classify the image produced by the point cloud.

Implementation. The key challenge in our implementation of LPC is how to determine the projection matrix for the hyperplane and the scale of each permutohedral lattice cell.

- *Projection matrix:* By referring to [1], we initialize the projection matrix as $\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$ and train the network in an end-to-end fashion. However, we observe that a big update of this matrix will make the training crash, and to avoid this issue, the update per iteration has to be tiny, leading to almost unnoticeable change eventually. The reason for this phenomenon is that this matrix has to satisfy certain requirements (see [1]), and thus the unconstrained update in backpropagation cannot work here. Therefore, in our experiments we initialize and fix the projection matrix. We refer to [13] for the lattice transformation.
- *Scale of lattice cell:* The parameter is simply not differentiable in backpropagation, and thus we tune it as a predefined hyper-parameter using cross-validation with grid search, same for the other hyper-parameters such as learning rate. Such scales have a significant impact on the image resolution used in 2D CNNs for classification.



(a) ModelNet40 (b) ScanNet

Figure 6: Training loss comparison on both datasets using EfficientNet-B5 as the backbone.

Specifically, we evaluate our LPC comprehensively based on three different CNNs, *i.e.,* VGG16 [45], ResNet50 [17], and EfficientNet-B5 [50] as the backbone network with randomly initialization. By default, the image resolution for each backbone network is 512×512, 128×128 and 456×456, respectively. On ModelNet40 [61] we train the three models using a learning rate of 10^{-4}, but on ScanNet [6] we only train our best model, *i.e.,* EfficientNet-B5, using a learning rate of 5×10^{-5}. We use Adam [22] as our optimizer in all of our experiments with weight decay of 10^{-4} and learning rate decay of 0.7 for every 20 epochs. Dropout [46] and data augmentation are applied as well when needed.

4. Experiments

Datasets. We conduct our experiments on ModelNet40 [61] and ScanNet [6]. ModelNet40 has a collection of 12,311 3D CAD objects from 40 common categories. It is split into 9,843 training and 2,468 test samples. Following [36, 62], we uniformly sample 1,024 points from the surface of the original point cloud per object and scale them into a unit ball. ScanNet contains 1,513 RGB-D scans from over 707 real indoor scenes with 2.5 million views. Following [26], we generate 12,445 training and 3,528 test point clouds from 17 categories, with 1,024 points for each point cloud as well.

Baselines. We compare our Lattice Point Classifier (LPC) with different adversarial defenders for point clouds that work with PointNet [36], including DUP-Net [75] (with SOR and the upsampling network), IF-Defense [60] (with ConvONet [35]), and robust pooling layer (RPL) [12]. We utilize public code [64] to train PointNet using the default setting and evaluate DUP-Net and IF-Defense based on the implementation in [60]. We report the best performance for each defender after fine-tuning. Specifically, we set $k = 2$ (the number of neighbor points) in KNN and $\alpha = 1.1$ (the percentage of outliers) for SOR, use 2 for the upsampling rate in DUP-Net, and choose the Welsch penalty function [7] for RPL [12] to replace the max pooling layer in PointNet. The model with RPL is trained with adversarial point clouds where 10% of input points are replaced by random outliers.

Adversarial attackers. Eight attackers are utilized to evaluate the robustness of different point cloud classifiers, including untargeted attackers (FGSM [11] and JGBA [31]), targeted attackers (perturbation, add, cluster and object attackers [62]), isometry transformation based attackers [72] (the untargeted black-box TSI attack and the targeted white-box CTRI attack), and GAN based LG-GAN attacker [74].

We apply both FGSM and JGBA to the full test set of both datasets. We slightly modify FGSM for attacking DUP-Net and IF-Defense under the white-box setting as both defenders are non-differentiable. To do so, during an attack we simulate the SOR process and obtain the indices of the remaining points based on which the gradient is passed to the remaining points. By default, the parameter ϵ in FGSM is set to 0.1, and the perturbation norm constraint ϵ, number of iterations n and step size α in JGBA are set to 0.1, 40, 0.01, respectively. Such parameters work well in practice.

For the targeted attackers, by following [62] we pick 10 large classes from ModelNet40, where a batch of 6 point clouds per class is randomly selected and attacked using the other 9 classes as the targets, leading to $10 \times 9 \times 6 = 540$ victim-target pairs. Similarly, from ScanNet we randomly select a batch of 6 point clouds as well from the 7 classes that contain more than 100 point clouds in test data. The learning rate of all the targeted attackers is set to 0.01. For DUP-Net and IF-Defense, we first attack clean PointNet to obtain adversarial point clouds, and then feed them into DUP-Net and IF-Defense for prediction. The perturbation attack first introduces small random perturbations on all original points, and uses L_2 norm to constrain the adversarial shifts. Using the add attacker, we add 60 points to each cloud and use Chamfer distance as the metric. Cluster and object attackers generate the initial clusters with parameter $\epsilon = 0.11$ in DBSCAN [9]. Using the cluster attacker, we add 3 clusters of 32 points to the original clouds. Using the object attacker, three 64-point adversarial objects are attached to each original point cloud.

For the TSI/CTRI attacker in [72], we evaluate the performance of LPC on ModelNet40 using EfficientNet-B5. By following [72] we use this attacker with the default settings to attack 2,000 randomly selected point clouds. The attacker applies the black-box TSI attacker first to trick the models, and then the white-box CTRI attacker if TSI fails.

We implemented a vanilla version of LPC in TensorFlow with binarized weights, ResNet50 as the backbone and 128×128 as the 2D image size. On ModelNet40, we trained a TensorFlow LPC model and integrated it into LG-GAN to generate adversarial samples. For the benchmark models, we trained a TensorFlow PointNet [36] model and generates adversarial samples with it since all other benchmark defenses are PointNet based. We followed the setups in [74] and set weight factor $\alpha = 100$.

Table 1: Our learning choice comparison in terms of test accuracy (%) with learning rate 10^{-4}.

T-Net [36]		✓			✓	
Binarized weights			✓		✓	✓
Random rotation				✓		✓
ResNet-50 on ModelNet40	88.2	87.3	**88.9**	60.0	88.2	75.2
EfficientNet-B5 on ModelNet40	-	-	**89.5**	83.3	-	84.8
EfficientNet-B5 on ScanNet	80.5	-	80.0	**82.6**	-	-

Table 2: Inference Time (ms) on an NVIDIA V100S GPU with batch size 1 (clean PointNet [36] takes 8.6 ms)

DUP-Net [75]	IF-Defense [60]	RPL [12]	LPC w/ VGG	LPC w/ ResNet	LPC w/ EfficientNet
808.1	1793.6	62.0	21.4	25.1	59.0

4.1. Ablation Study

Learning choices. In Table 1 we list three learning choices that we would like to evaluate for improving the performance of vanilla LPC, *i.e.,* T-Net used in PointNet, binarized barycentric weights, and random rotation in data augmentation (together with point cloud random shifting and dropping [36]). We can see that: (1) T-Net seems to deteriorate the performance always. (2) Binarized weights work better on ModelNet40 than ScanNet, but compared with using barycentric weights the difference is <1%. (3) Random rotation improves the performance on ScanNet, but worsens it on ModelNet40. This phenomenon can be partially explained from the training loss curves as shown in Fig. 6, where on ScanNet the overfitting occurs clearly without random rotation while on ModelNet40 random rotation makes the convergence slower and unstable.

Running time. Overall LPC achieves the fastest inference speed among all competing defenses as shown in Table 2. Our deepest model (17fps) is over 13 times faster than SOR based defenses. We can also improve the running time using shallower networks such as VGG16 (45fps) by slightly sacrificing the performance. We also tested the running time for each key component (declarative node and backbone 2D networks). With EfficientNet-B5 as the backbone, the declarative node and 2D network take 17.1 and 41.9 ms respectively during inference, and 15.5 and 165.3 ms during backpropagation. Clearly, the gradient passing through the declarative node takes relatively constant time in both feedforward and backpropagation. Considering the depth of EfficientNet-B5, the time spent on the declarative node is actually pretty long (recall that there is no learnable parameter inside), especially during inference. This partially validates our intuition of defending the adversarial attacks using implicit gradients.

Table 3: Test accuracy (%) comparison (higher is better) on the full test datasets, where "-" indicates no result.

		PointNet	PointNet w/ DUP-Net [75]	PointNet w/ IF-Defense [60]	PointNet w/ RPL [12]	LPC w/ VGG16	LPC w/ ResNet50	LPC w/ EfficientNet
ModelNet40	No attack	**90.15**	89.30	87.60	84.76	88.65	88.90	89.51
	FGSM [11]	45.99	61.63	38.75	0.04	88.65	88.90	**89.51**
	JGBA [31]	0.00	1.14	5.37	0.00	88.65	88.90	**89.51**
ScanNet	No attack	**84.61**	83.62	80.19	76.02	-	-	83.16
	FGSM [11]	45.66	73.67	71.14	1.70	-	-	**83.16**
	JGBA [31]	0.00	7.77	13.45	0.00	-	-	**83.16**

Table 4: Attack success rate (%) comparison (lower is better), where "-" indicates no result. The standard deviations of our LPC range from 0% to 0.28% in success rate on ModelNet40.

		PointNet	PointNet w/ DUP-Net [75]	PointNet w/ IF-Defense [60]	PointNet w/ RPL [12]	LPC w/ VGG16	LPC w/ ResNet50	LPC w/ EfficientNet
ModelNet40	FGSM [11]	48.99	30.77	55.78	99.95	**0.00**	**0.00**	**0.00**
	JGBA [31]	100.00	98.73	93.85	100.00	**0.00**	**0.00**	**0.00**
	Perturbation [62]	100.00	**0.095**	**0.095**	3.95	0.56	0.37	0.38
	Add [62]	99.72	**0.095**	**0.095**	3.33	0.56	0.19	0.19
	Cluster [62]	98.34	6.76	6.30	17.04	1.11	**0.93**	1.21
	Object [62]	98.43	1.02	1.11	74.07	0.93	1.11	**0.75**
ScanNet	FGSM [11]	46.03	10.29	11.28	97.76	-	-	**0.00**
	JGBA [31]	100.00	90.55	83.21	100.00	-	-	**0.00**
	Perturbation [62]	100.00	3.17	**2.38**	3.03	-	-	12.70
	Add [62]	100.00	**1.98**	2.38	18.65	-	-	2.78
	Cluster [62]	100.00	30.16	23.81	40.87	-	-	**9.92**
	Object [62]	100.00	7.14	7.54	85.71	-	-	**5.95**

Table 5: Performance comparisons under LG-GAN [74] attacks. Attack success rate (Succ., %) on different models and their classification accuracy (Accu., %) under attack

	PointNet [36]	DUP-Net [75]	IF-Defense [60]	RPL [12]	LPC w/ ResNet
Accu.	0.6	31.6	37.2	56.8	**76.7**
Succ.	97.0	13.9	10.4	2.6	**0.8**

4.2. State-of-the-art Performance Comparison

We measure the robustness of classifiers using two metrics, *i.e.,* classification accuracy, and attack success rate. Classification accuracy under attacks is measured by feeding the entire adversarial attacked test set to the victim models. It is only calculated for untargeted attackers. The attack success rate is the ratio between the number of successful attacks to the number of all attempt attacks. For untargeted attacks, tricking the model to predict a wrong class is considered a success, while for targeted attacks it will have to trick the victim model to a specific class to be considered as successful. Untargeted attackers will only attack the point clouds that are correctly classified by the victim models, and targeted attackers will attack victim-target pairs.

4.2.1 Classification Accuracy

We summarize our comparison in Table 3. On ScanNet, we only show the performance of LPC with EfficientNet-B5, because it achieves the best accuracy on ModelNet40. We can see that: (1) On the clean test data with no attack, PointNet outperforms all the robust classifiers by small margins. (2) Using both attackers, our LPC variants work consistently and significantly better than the other defender-based robust classifiers as well as vanilla PointNet by large margins. For instance, compared with PointNet with IF-Defense under the JGBA attacks, the performance gaps are 84.14% and 69.71% on ModelNet40 and ScanNet, respectively. (3) For the backbone networks in LPC, it seems that the difference in performance is small among different CNNs. Though more evaluations are needed to confirm this, it also demonstrates the robustness of our approach.

4.2.2 Attack Success Rate

Performance summary. We list our comparison in Table 4 and Table 5. We can see that: (1) Our LPC variants achieve perfect results under both FGSM and JGBA attacks. Notice that compared with the other attackers, these two attackers

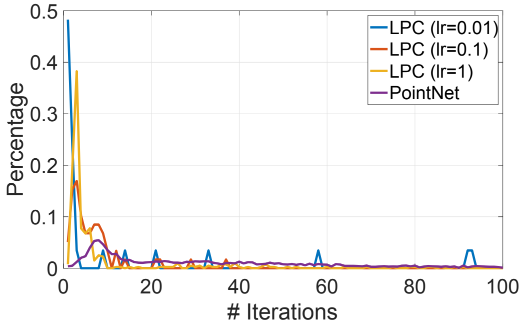


Figure 7: Distribution comparison of successful perturbation attacks on ModelNet40. To avoid the sparsity, LPC statistics are collected based on the cases from all the three models.

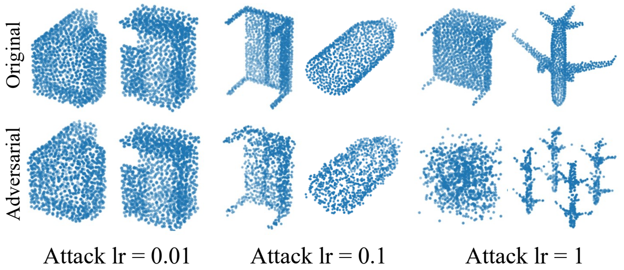


Figure 8: Successful adversarial perturbation examples on ModelNet40 for LPC with EfficientNet-B5.

aim to find adversarial examples fully based on gradients with no sampling. Their failure again strongly demonstrates the power of implicit gradients in defending gradient based adversarial attacks. (2) The SOR defense mechanism seems to work better for the perturbation and add attackers, but worse for the cluster and object attackers, compared with our LPC. However, overall LPC still achieves the best performance. (3) Our LPC also outperforms other defenses under GAN based attacks. Interestingly RPL [12] outperforms SOR defenses under LG-GAN attack, while under gradient based attacks the SOR methods are better. It implies LG-GAN has better transferability. The two SOR based defenses are gray-box attacks (partial access to the model). Gradient based adversarial samples suffer significant success rate drop as shown in Table 4. But LG-GAN transfers well to SOR based defenses, maintaining over 10% attack success rate.

Using the TSI/CTRI attacker [72], our LPC with EfficientNet-B5 achieves the same success rate for both TSI and CTRI attackers on ModelNet40. Without random rotation, the result is 99.54%, but with random rotation, the performance decreases to 67.33% which is significantly better than PointNet (99.50% and 99.55% for TSI and CTRI, respectively). Such results also demonstrate that random rotation to point clouds as a means of data augmentation improves not only accuracy but also model robustness.

Importance of gradients in finding successful adversarial examples for LPC. To attack each point cloud, the four targeted attackers in [62] conduct 10 random searches where 500 iterations are done to find the optimal adversarial samples. We notice that for PointNet, the best attacks could occur at any time within these 500 iterations, but for our LPC most optimal attacks happen at the first few iterations, as shown in Fig. 7 with different learning rates for the perturbation attacker. This behavior indicates that, in order to attack LPC, random search (a sampling based method) has contributed more to most of the successful attacks, rather than gradient-based iterations. With the increase of the learning rate, the gradients start to find more optimal adversarial samples. However, as we illustrate in Fig. 8, with larger learning rates, the adversarial samples will not look similar to the original point clouds. For instance, with learning rate of 1, the point cloud of airplane has been totally changed to a mixture of smaller airplane point clouds, which is not an adversarial attack anymore. To sum up, such analysis again demonstrates the great potential of implicit gradients in defending adversarial attacks.

5. Conclusion

In this paper, we aim to address the problem of robust 3D point cloud classification by proposing a family of novel robust structured declarative classifiers, where a declarative node is defined by a constrained optimization problem such as the reconstruction of point clouds. The key insight in our approach is that the implicit gradients through the declarative node can help defend the adversarial attacks by leading them to wrong updating directions for inputs. We formulate the learning of our classifiers based on bilevel optimization, and further propose an effective and efficient instantiation, namely, Lattice Point Classifier (LPC). The declarative node in LPC is defined as structured sparse coding in the permutohedral lattice, whose outputs, *i.e.,* barycentric weights, are further transformed into images for classification using 2D CNNs. LPC is end-to-end trainable, and achieves state-of-the-art performance on robust classification on ModelNet40 and ScanNet using seven different adversarial attackers.

Limitations. Currently the projection in the permutohedral lattice transformation in LPC is not learned but simply fixed as initialization. Also more evaluations for demonstrating the robustness of our approach across different backbone networks and datasets are desirable. Therefore, in our future work we will investigate more on how to properly learn the projection with its physical conditions and conduct more experiments to further demonstrate our robustness.

Acknowledgement

K. Li and C. Zhong were supported in part by USDA NIFA under the award no. 2019-67021-28996. Z. Zhang was supported in part by NSF grant CCF-2006738. G. Wang was partly supported by NSERC grant RGPIN-2021-04244.

References

[1] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer graphics forum*, volume 29, pages 753–762. Wiley Online Library, 2010. 2, 3, 5

[2] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in Neural Information Processing Systems*, 32:9562–9574, 2019. 2

[3] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017. 2

[4] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018. 1

[5] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1):25–45, 2021. 2, 3

[6] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 5

[7] John E Dennis Jr and Roy E Welsch. Techniques for nonlinear least squares and robust regression. *Communications in Statistics-simulation and Computation*, 7(4):345–359, 1978. 5

[8] Xiaoyi Dong, Dongdong Chen, Hang Zhou, Gang Hua, Weiming Zhang, and Nenghai Yu. Self-robust 3d point recognition via gather-vector guidance. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11513–11521. IEEE, 2020. 3

[9] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. 6

[10] Kexue Fu, Shaolei Liu, Xiaoyuan Luo, and Manning Wang. Robust point cloud registration framework based on deep graph matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8893–8902, 2021. 2

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 2, 6, 7

[12] Stephen Gould, Richard Hartley, and Dylan John Campbell. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2, 5, 6, 7, 8

[13] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3254–3263, 2019. 2, 3, 5

[14] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR, 2019. 1

[15] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 2

[16] Abdullah Hamdi, Sara Rojas, Ali Thabet, and Bernard Ghanem. Advpc: Transferable adversarial perturbations on 3d point clouds. In *European Conference on Computer Vision*, pages 241–257. Springer, 2020. 3

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[18] Varun Jampani, Martin Kiefel, and Peter V Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4452–4461, 2016. 3

[19] Anil Kag, Ziming Zhang, and Venkatesh Saligrama. Rnns incrementally evolving on an equilibrium manifold: A panacea for vanishing and exploding gradients? In *International Conference on Learning Representations*, 2019. 2

[20] Sofia Karygianni and Pascal Frossard. Structured sparse coding for image denoising or pattern detection. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3533–3537. IEEE, 2014. 2

[21] Jaeyeon Kim, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Minimal adversarial examples for deep learning on 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7797–7806, 2021. 2

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[23] Loic Landrieu and Mohamed Boussaha. Point cloud oversegmentation with graph-structured deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7440–7449, 2019. 2

[24] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018. 2

[25] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019. 2

[26] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018. 2, 5

[27] Daniel Liu, Ronald Yu, and Hao Su. Extending adversarial attacks and defenses to deep 3d point cloud classifiers. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2279–2283. IEEE, 2019. 2

[28] Daniel Liu, Ronald Yu, and Hao Su. Adversarial shape perturbations on 3d point clouds. In *European Conference on Computer Vision*, pages 88–104. Springer, 2020. 1

[29] Hongbin Liu, Jinyuan Jia, and Neil Zhenqiang Gong. Point-

guard: Provably robust 3d point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6186–6195, 2021. 1, 3

[30] Yecheng Lyu, Xinming Huang, and Ziming Zhang. Learning to segment 3d point clouds in 2d image space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12255–12264, 2020. 1, 2

[31] Chengcheng Ma, Weiliang Meng, Baoyuan Wu, Shibiao Xu, and Xiaopeng Zhang. Efficient joint gradient based attack against sor defense for 3d point cloud classification. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1819–1827, 2020. 1, 2, 3, 6, 7

[32] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 2

[33] Xi Mo, Usman Sajid, and Guanghui Wang. Stereo frustums: a siamese pipeline for 3d object detection. *Journal of Intelligent & Robotic Systems*, 101(1):1–15, 2021. 2

[34] Mesut Ozdag. Adversarial attacks and defenses against deep neural networks: a survey. *Procedia Computer Science*, 140:152–161, 2018. 2, 3

[35] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 523–540. Springer, 2020. 5

[36] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 2, 5, 6, 7

[37] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016. 2

[38] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 2

[39] Guocheng Qian, Abdulellah Abualshour, Guohao Li, Ali Thabet, and Bernard Ghanem. Pu-gcn: Point cloud upsampling using graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11683–11692, 2021. 2

[40] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, volume 32, pages 113–124, 2019. 1, 2

[41] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586, 2017. 2

[42] Radu Alexandru Rosu, Peer Schütt, Jan Quenzel, and Sven Behnke. Latticenet: fast spatio-temporal point cloud segmentation using permutohedral lattices. *Autonomous Robots*, pages 1–16, 2021. 3

[43] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, 2008. 3

[44] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719, 2020. 2

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 5

[47] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2530–2539, 2018. 2, 3

[48] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 2

[49] Arthur Szlam, Karol Gregor, and Yann LeCun. Fast approximations to structured sparse coding and applications to object classification. In *European Conference on Computer Vision*, pages 200–213. Springer, 2012. 2

[50] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 5

[51] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017. 3

[52] Tzungyu Tsai, Kaichen Yang, Tsung-Yi Ho, and Yier Jin. Robust adversarial objects against deep learning models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 954–962, 2020. 3

[53] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13716–13725, 2020. 3

[54] Chengyu Wang, Jia Wang, and Qiuzhen Lin. Adversarial attacks and defenses in deep learning: A survey. In *International Conference on Intelligent Computing*, pages 450–461. Springer, 2021. 2, 3

[55] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11, 2017. 2

[56] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 1, 2

[57] Wei Wei, Lei Zhang, Chunna Tian, Antonio Plaza, and Yanning Zhang. Structured sparse coding-based hyperspectral imagery denoising with intracluster filtering. *IEEE Transac-*

tions on Geoscience and Remote Sensing, 55(12):6860–6876, 2017. 2

[58] Matthew Wicker and Marta Kwiatkowska. Robustness of 3d deep learning in an adversarial setting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11767–11775, 2019. 1, 2, 3

[59] Yuanwei Wu, Tim Marks, Anoop Cherian, Siheng Chen, Chen Feng, Guanghui Wang, and Alan Sullivan. Unsupervised joint 3d object model learning and 6d pose estimation for depth-based instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 1–10, 2019. 2

[60] Ziyi Wu, Yueqi Duan, He Wang, Qingnan Fan, and Leonidas J Guibas. If-defense: 3d adversarial point cloud defense via implicit function based restoration. *arXiv preprint arXiv:2010.05272*, 2020. 1, 3, 5, 6, 7

[61] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 1, 2, 5

[62] Chong Xiang, Charles R Qi, and Bo Li. Generating 3d adversarial point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9136–9144, 2019. 1, 2, 5, 6, 7, 8

[63] Zhen Xiang, David J. Miller, Siheng Chen, Xi Li, and George Kesidis. A backdoor attack against 3d point cloud classifiers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7597–7607, October 2021. 1, 3

[64] Xu Yan. Pointnet/pointnet++ pytorch. *https://github.com/yanx27/Pointnet_Pointnet2_pytorch*, 2019. 5

[65] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5589–5598, 2020. 2

[66] Jiancheng Yang, Qiang Zhang, Rongyao Fang, Bingbing Ni, Jinxian Liu, and Qi Tian. Adversarial attack and defense on point sets. *arXiv preprint arXiv:1902.10899*, 2019. 2, 3

[67] Ze Yang and Liwei Wang. Learning relationships for multi-view 3d object recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7505–7514, 2019. 2

[68] Tan Yu, Jingjing Meng, and Junsong Yuan. Multi-view harmonized bilinear network for 3d object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 186–194, 2018. 2

[69] Jinlai Zhang, Lyujie Chen, Bo Ouyang, Binbin Liu, Jihong Zhu, Yujing Chen, Yanmei Meng, and Danfeng Wu. Pointcutmix: Regularization strategy for point cloud classification. *arXiv preprint arXiv:2101.01461*, 2021. 3

[70] Jinlai Zhang, Binbin Liu, Lyvjie Chen, Bo Ouyang, Jihong Zhu, Minchi Kuang, Houqing Wang, and Yanmei Meng. The art of defense: letting networks fool the attacker. *arXiv preprint arXiv:2104.02963*, 2021. 3

[71] Zheng Zhang, Yong Xu, Jian Yang, Xuelong Li, and David Zhang. A survey of sparse representation: algorithms and applications. *IEEE access*, 3:490–530, 2015. 3

[72] Yue Zhao, Yuwei Wu, Caihua Chen, and Andrew Lim. On isometry robustness of deep 3d point cloud models under adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1201–1210, 2020. 3, 6, 8

[73] Tianhang Zheng, Changyou Chen, Junsong Yuan, Bo Li, and Kui Ren. Pointcloud saliency maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1598–1606, 2019. 3

[74] Hang Zhou, Dongdong Chen, Jing Liao, Kejiang Chen, Xiaoyi Dong, Kunlin Liu, Weiming Zhang, Gang Hua, and Nenghai Yu. Lg-gan: Label guided adversarial network for flexible targeted attack of point cloud based deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10356–10365, 2020. 1, 3, 6, 7

[75] Hang Zhou, Kejiang Chen, Weiming Zhang, Han Fang, Wenbo Zhou, and Nenghai Yu. Dup-net: Denoiser and upsampler network for 3d adversarial point clouds defense. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1961–1970, 2019. 1, 3, 5, 6, 7