Self-Organizing and Parallel-Process Driven Fast Generation of Adversarial Examples for 3D Point Clouds

ABSTRACT

As an emerging threat to point cloud deep learnings, adversarial example attacks have received increasing attentions since they can easily fool the target systems and then lead to serious damages. Different to conventional time- and resource-consuming methods, in this paper we propose a new data-driven method to fast generate adversarial examples on 3D point cloud deep learnings. Based on self-organizing map, we first design a global feature extraction network to process point cloud input with group coding, which can help to speed up the coding process and reduce the resource usage. We then use a dual-process point cloud restoration network to generate point clouds with adversarial properties. Designed with two parallel branches, the restoration network can quickly obtain effective adversarial examples under the premise of low resource usage. Benchmark-based experiments are conducted to evaluate the efficiency of our method. Specifically, we can achieve up to 511% faster with only 34.19% GPU resource usage comparing with existing methods. In addition, our method increases up to 12% in the success rate of attacking unseen networks (transferability) and 9.8% in the ability to break through point cloud networks with defensive policies.

KEYWORDS

Deep Neural Networks, 3D Point Clouds, Adversarial Examples

1 INTRODUCTION

Deep Neural Networks (DNNs) have achieved impressive success in perception tasks in the fields of autonomous driving, robotics, and virtual reality. However, it has been found that DNNs are vulnerable to adversarial attacks, which can produce imperceptible disturbances to the input and lead the target system to misbehave. This type of research has made significant progress on 2D images [8, 14, 19, 20], and a plethora of works have proved the effectiveness of adversarial example attacks on 2D images [3]. With the combination of deep learnings and 3D sensors like lidar [2, 12, 18, 21, 22], point cloud DNNs are increasingly utilized in a variety of safety-critical applications [1, 13], such as unmanned driving and security monitoring, etc. Unfortunately, point cloud DNNs are also seriously threatened by adversarial example attacks, which will definitely increase the risk in safety-critical scenarios, leading to serious safety accidents.

Recently, there are certain studies on adversarial example attacks on point cloud deep learnings. The first method is to obtain adversarial examples by adding adversarial points or perturbing key points [18]. After that, in order to make the adversarial examples exemplified as real objects, K-Nearest Neighbor (KNN) loss leveraged to smooth the generated point cloud adversarial examples, so that the attack is physically achievable[12]. In addition, AdvPC is proposed to evaluate the feasibility to generate adversarial examples by focusing on the transferability of point clouds [2]. Although the above-mentioned methods have good results, the generation process is very slow and consumes a lot of hardware resources.

In contrast, we propose a fast generation of adversarial examples for 3D Point Clouds (FADPC). FADPC can generate adversarial examples quickly and efficiently. Our method mainly consists of a Self-Organizing Map (SOM) based global feature extraction network and a dual-process based point cloud restoration network. The global feature extraction network uses SOM to group the original point clouds, and then group to extract global features and merge them to achieve the purpose of quickly and effectively encoding the point clouds into global features. The dual-process based point cloud restoration network uses the convolution branch and the MLP branch to obtain the point clouds respectively, and then merge them to obtain the final adversarial examples. Doing so can quickly restore the global features of the point cloud to a 3D point cloud. The adversarial loss is added in the restoration process, so that the restored 3D point cloud can fool the classifier. Our method seldom depends on the victim network and can be better generalized to different networks. In summary, this paper has the following contributions:

- We design a global feature extraction network. The method based on SOM grouping extraction can increase the speed of point cloud global feature extraction and reduce resource consumption.
- We design a dual-process based point cloud restoration network, including a process supported by a small-scale fully connected structure and a process with a convolution module as the backbone. This method effectively improves the restoration speed of the point cloud and reduces the resource usage.
- Benchmark-based experiments on existing methods are conducted to evaluate the superiority of our method, which can achieve up to 511% faster speed with 65.81% lower resource usage while ensuring the success rates of attacks. Moreover, our method can obtain better performance in the aspects of transferability and breaking through on defensive methods.

2 RELATED WORK

2.1 Deep Learning on Point Clouds

PointNet [9] is the first DNN work that directly uses 3D point cloud data. It uses max pooling to aggregate the features of each point into a global descriptor vector. PointNet's order of the input points is unchanged, because the feature extraction of each point is same, and the maximum pooling operation is the same. PointNet++ [10] is an upgrade of PointNet, which divides points into several groups of different levels so that features from multiple 3D scales can be extracted hierarchically. The closest approach is to run convolution among the neighbors of a point, instead of using a point-by-point operation [4, 5, 7, 11, 15]. In contrast to PointNet and its variants, these works are more focused on the extraction of local features to

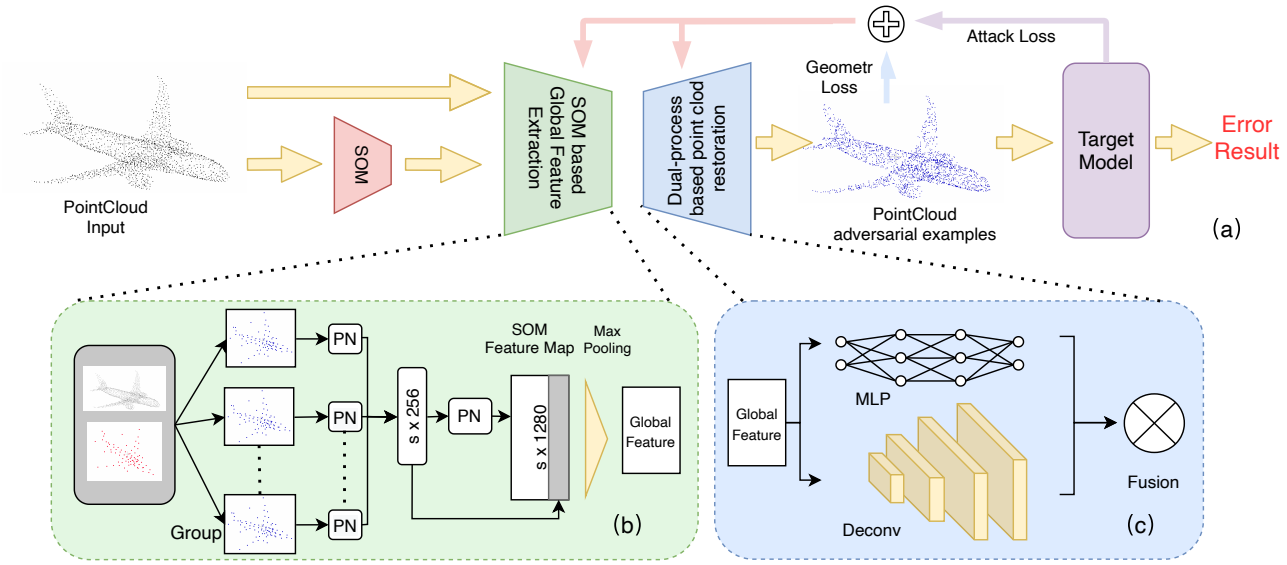


Figure 1: Design Framework of FADPC

achieve recognition results. In this paper, we choose three point-by-point processing networks, i.e., PointNet, PointNet++ single-scale (SSG), PointNet++ multi-scale (MSG) and Dynamic Graph Convolutional Networks (DGCNN) [16], to verify the feasibility and performance of our method. We will study the effectiveness of our method on these three networks, the transferability between the three networks, and the speed and resource consumption of our method to generate adversarial examples with these three networks as the target networks.

2.2 Adversarial Attacks on Point Clouds

The first method was proposed by Xiang et al [18], which deployed two attack methods: perturbating against key points and adding perturbation points. Unfortunately, these methods are easy to be defended by simple statistical methods, and basically do not have transferability over other networks. Cai et al. proposed to add KNN loss to the generated points [12] to make the point cloud generated by this point cloud attack method smoother, so that the generated point cloud can be restored to real space or even be 3D printed. Recently, Abdullah et al. proposed the AdvPC [2] which generates a point cloud disturbance by adding a transferability loss during the autoencoder training process, and then it adds the point cloud disturbance to the original point cloud to obtain a highly transferable adversarial examples. In contrast, our method is to directly generate point cloud adversarial examples without adding any perturbations.

2.3 Defenses Against Point Cloud Attacks

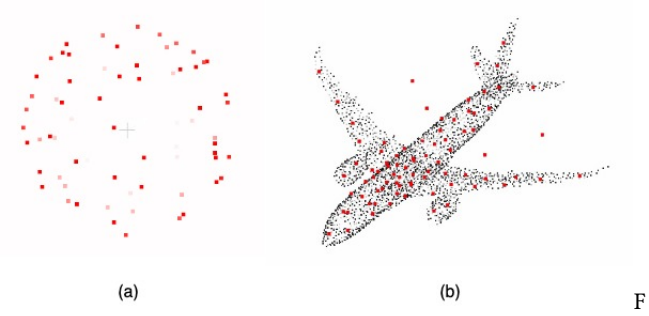
Zhou et al. proposed a method based on Statistical Outlier Removal (SOR) to defend against point cloud adversarial example attacks. SOR uses KNN to identify and remove abnormal points in the point cloud to achieve defense effect. Zhou et al. also proposed DUP-Net [24], which combines SOR with the point cloud up-sampling network PU-Net [23] to further strengthen the defense performance. They also proposed to remove unnatural points through Simple Random Sampling (SRS), where each point has the same probability of being randomly removed to achieve the purpose of defense. Moreover, Xiang et al. [18] also proposed adversarial training on the attacked point cloud as a defense mode.

3 APPROACH

The design framework of FADPC is demonstrated in Figure 1. The point cloud samples are trained by FADPC to generate adversarial examples for 3D point clouds, which can help to reduce the dependence of the attack on the target network. In this section, we first introduce the self-organizing input process. Then we show the main sub-networks of our model: SOM based global feature extraction network and dual-process based point cloud restoration network. Finally the training loss design is presented.

3.1 Self-Organizing Input Preprocessing

Our work leverages SOM to assist in point cloud grouping. SOM can generate a low-dimensional discrete map by learning the data in the input space[6]. We construct an initial SOM with s nodes. The training method of SOM is different from general neural network training based on the reverse transfer of loss function. It uses a competitive learning strategy, relying on neurons to compete with each other gradually. And a neighborhood function is used to maintain the topological structure of the input space. Because the input of point cloud is normalized within [-1, 1], we distribute the initial nodes of the SOM uniformly in a unit sphere with a radius of 1. Since the point cloud is unordered as input, in order to make the SOM training results of the same point cloud consistent, we perform a unified SOM updated after calculating the influence of all points. This batch calculation method can make the arrangement unchanged. We use this method to quickly generate the corresponding SOM for each 3D point cloud model. An illustrative example of SOM initial state and the training result is shown in Figure 2 (b).



F

Figure 2: (a): Initial state of SOM. (b): Training result of SOM.

3.2 SOM based Global Feature Extraction

Our SOM based global feature extraction network has two inputs: the original point cloud and its corresponding SOM. As shown in the group operation in Figure 1(b), SOM is the index of the sample point cloud. As shown in Eq. 1, we take each point m_j on the SOM as the starting point to sequentially perform Farthest Point Sampling (FPS) on the point cloud p_i, and divide the point cloud into s groups, each with n points.

$$p_{ij} = FPS\left(p_i | m_j, j = 0, 1, \ldots, s-1\right) \tag{1}$$

It can be seen from the grouping results in the Figure 1(b) that the sampling distribution of each group is very uniform and can well cover the sampling space. Then we normalize each point p_{ij} by its associated node m_i, shown as follows:

$$p_{ij} = m_j - p_{ij} \tag{2}$$

As shown in Figure 1(b), after the point cloud is grouped and normalized, the points are sent to the PointNet (PN) coding module for feature extraction. The PN module is composed of a fully connected layer l and a maximum pooling layer. Among them, ψ is a nonlinear activation function, and the output of the fully connected layer in PN is shown in Eq. 3.

$$p_{ij}^{l+1} = \psi\left(V^l p_{ij}^l + b^l\right) \tag{3}$$

We perform the maximum pooling operation on the single point features obtained by the fully connected layer, aggregate the features on the SOM node corresponding to each subgraph, and obtain each node feature, which is shown in Eq. 4. All the node features together form a node feature graph.

$$m_j^0 = \max\left(p_{ij}, i = 0, 1, \ldots, n-1\right) \tag{4}$$

We send the node feature map to the PN module to get the global feature. The global features are spliced with the node feature map to obtain the SOM feature map of the point cloud, which is further used to represent the input point cloud. Finally, the SOM feature map is pooled to the maximum and aggregated to obtain the global feature.

The point cloud passes through one SOM and two batches of PN modules before getting the final global features. Because the process of obtaining the point cloud SOM and the search process of FPS are determined and the PN module is also constant, so we can guarantee the node characteristics and the global all features are permutation invariance. Because SOM can well reveal the spatial distribution of point clouds, the SOM node is used as the starting point of the FPS algorithm. We use it to sample from the original point cloud, and each set of mini point clouds obtained can well represent the global characteristics.

Compared with only using the PN module for global feature extraction, SOM group extraction does not need to abstract the dimensions of all points to a very high dimension. It only needs to group all points to a SOM node, and then aggregate the SOM node features to a high dimension. Therefore, the use of SOM speeds up the extraction of global features and reduces resource occupation. The global features are obtained from the maximum pooling of the SOM feature map, which is composed of global features and the SOM node feature splicing. The SOM node features can complement each other to make the global information more detailed. The global feature is obtained by maximum pooling after stitching the node feature map and the global feature, which can also improve the effect of subsequent point cloud restoration.

3.3 Dual-Process based Point Cloud Restoration

We design a point cloud restoration network that can recover the input point cloud from the global characteristics of the point cloud. In the past, the most commonly used method is to connect a series of fully connected layers after inputting the global features, and finally generate a 3-dimension vector of length N. However, if there are enough points generated (N is large enough), the parameters of the model will become very large. Therefore, the model operation will take up a lot of resources, and the point cloud generation speed will be very slow.

As shown in Figure 1(c), the restoration network is designed with two parallel branches, namely MLP branch and convolutional branch. The MLP branch is composed of fully connected layers. The MLP branch continues to abstract the global features of the point cloud restoration network input to a higher dimension, and then reshape it into an $N \times 3$ style output. Each point output by the fully connected layer is independently predicted and has a high degree of flexibility. Due to a large number of parameters, the training speed is too slow and it is not suitable for all points to be generated by the fully connected layer.

Refering to many point cloud generation and depth estimation methods, we construct a special convolution branch. Our convolution branch consists of the deconv module, which has two design parts: interpolation and convolution layers. First, interpolation is used to transfer the rough features of the previous layer, and then the output result is obtained through a 3×3 convolutional layer. The convolutional space has continuity, so the point cloud part generated by the convolution branch has better geometric consistency. Moreover, the convolution branch parameters are less and the training speed is faster. The two branch generation results together constitute the final adversarial examples.

3.4 Hybrid Loss

To achieve high-level reconstruction accuracy and attack success rate, we use a hybrid loss composed of attack loss and geometry loss.

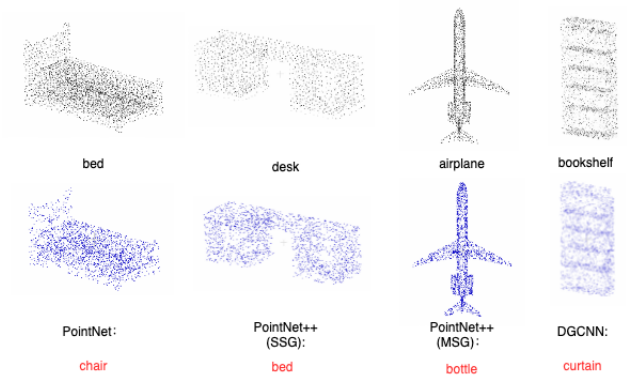


Figure 3: Examples of adversarial examples generated by FADPC. The black point cloud is the original point cloud, and the blue point cloud is the adversarial example. The red label is the misclassification result of the adversarial examples under the target network.

3.4.1 Attack Loss. The adversarial examples generated by our method are to make the target network misclassify. Inspired by the method from 2D adversarial examples, we design an adversarial loss. The first part of the loss is to reduce the target model's confidence in the true label. We use the estimated probability of the model as the direct loss of the attack. As expressed in Eq. 5, k is the type of sample, g is the coding value of the target model, and $\hat{g}$ is the predicted probability of the target model.

$$L_1 = \sum_k g_k \hat{g}_k \tag{5}$$

The second part of the loss is the misleading target model, which is expressed in Eq. 6. We choose the error category with the highest confidence of the target model for the current generated sample as the explicit direction of overall optimization.

$$L_2 = 1 - \max_k (1 - g_k)\, \hat{g}_k \tag{6}$$

3.4.2 Geometry Loss. There are many ways to express the structural differences between two point clouds. We use structure loss consisting of L_2 distance, chamfer distance, and hausdoff distance to make up for its shortcomings. Given two point set X and Y, L_2 distance, chamfer distance , and hausdoff distance can be expressed as:

$$D_{L_2} = \sum_{i=1}^{X} \|X_i - Y_i\|_2^2 \tag{7}$$

$$\begin{aligned} D_{chamfer}(X,Y) = &\frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} \|x - y\|_2^2 \\ &+ \frac{1}{|Y|} \sum_{x \in Y} \min_{y \in X} \|y - x\|_2^2 \end{aligned} \tag{8}$$

$$D_{hausdorff}(X,Y) = \max_{x \in X} \min_{y \in Y} \|x - y\|_2^2 \tag{9}$$

L_2 distance only calculates the distance between pairs of points, so the calculation speed is very fast. But because it lacks the information of points to surrounding points, treating this loss as the only loss will lead to the entire network training abnormal. So we add chamfer distance and hausdoff distance. The hausdoff distance traverses the entire point set to find the closest point in the original point cloud for the points in the counterpoint cloud, and outputs the maximum squared distance of these point pairs. chamfer distance is a measurement method similar to hausdoff distance. The difference is that chamfer distance takes the average of all nearest pairs, while Hausdor distance takes the maximum value. The chamfer and Hausdor distance are not directly calculated on the distance between point pairs but based on two complete point sets, so the original shape of the point cloud can be preserved as much as possible. Figure 3 shows four instances of adversarial examples generated by our method.

4 EXPERIMENT

In this paper, we use ModelNet40 [17] , a widely used 3D point cloud dataset, to test effectiveness of the proposed adversarial examples generation method. The ModelNet40 data set contains 12,311 CAD models in 40 different categories. 9,843 objects were used for training, and another 2,468 were used for testing. We uniformly sample 10,000 points from the surface of each object and normalize them to a unit sphere. The sample distribution of the ModelNet40 data set is very uneven. We select 10 classes with the most samples from 40 classes, and randomly select 50 examples from each class sample to generate point cloud adversarial examples, so we have 500 samples used in our experiments. We use PointNet [9], PointNet++ in single-scale (SSG) and multi-scale (MSG) [10], and DGCNN [16] as the target classification network, and use the default settings to train the model. We compare FADPC against the state-of-the-art baselines AdvPC[2], 3D-Adv [18] and KNN Attack [12]. In our experiments, all attacks were untargeted attacks. We use chamfer distance as a metric to generate standard attacks and have measure the success rate of attacks under the budgets of chamfer distance=0.10. This threshold is chosen because it enables the attack to achieve a 100% success rate on the target network and the possibility of transferring to other networks. Under this threshold, the high similarity between the generated adversarial examples and the original can also be guaranteed.

All experiments are implemented using PyTorch on NVIDIA RTX2080Ti. In most experiments, we use SOM with 81 nodes. We use Adam to optimize the network, where the batchsize is set to 10 and the point cloud is downsampled to 2048 points as input. Batch-normalization and ReLU activation are applied to every layer. The performances of FADPC are evaluated in three aspects, i.e., attack performance and transferability, the ability to break through defensive methods, and the speed of adversarial example generation.

4.1 Attack Transferability

To evaluate the transferability, we use the percentage of adversarial examples misclassified by the target network as an attack performance indicator. We optimize the adversarial examples for each target network, and take them as the input of other DNNs to test

Table 1: Comparison of attack success rate on target networks and unseen networks.

Target Model	PointNet				PointNet++ (MSG)				PointNet++ (SSG)				DGCNN			
Attack	3D-Adv	KNN	AdvPC	**FADPC**	3D-Adv	KNN	AdvPC	**FADPC**	3D-Adv	KNN	AdvPC	**FADPC**	3D-Adv	KNN	AdvPC	**FADPC**
PointNet	100	100	98.0	**100**	6.8	7.2	13.2	**15.7**	6.8	6.8	10.8	**12.8**	10.8	7.2	20.8	**31.2**
PointNet++ (MSG)	8.4	9.2	17.2	**26.7**	100	100	94.8	**98.4**	8.8	8.8	27.6	**31.3**	14.4	11.2	32.4	**47.8**
PointNet++ (SSG)	8.8	8.4	28.0	**33.2**	32.4	22.8	53.2	**66.3**	100	100	96.4	**98.7**	39.6	13.6	52.4	**58.3**
DGCNN	7.2	7.2	22.0	**24.6**	14.8	8.4	33.2	**46.1**	8.0	7.6	26.8	**29.9**	100	100	85.2	**94.7**

the success rate of attacks. In this experiment, no defensive strategy is implemented in the original point cloud DNNs.

Table 1 shows the attack effect of FADPC and the baseline methods. First, the success rate of FADPC attacks is basically the same as other existing methods. FADPC attacks on PointNet can guarantee a 100% success rate. The success rate of FADPC in attacking PointNet++ and DGCNN is also close to 100%. In specific, the attack rates of FADPC are 1.6% lower in attacking PointNet++ (MSG), 1.3% lower in attacking PointNet++ (SSG) and 5.3% lower in attacking DGCNN than 3D-Adv and KNN. This is because 3D-Adv and KNN can utilize network parameters to generate adversarial examples, while FADPC has no any knowledge of target network parameters. Compared with AdvPC, which also does not need to read network parameters, FADPC can have better performance in all situations.

Table 1 also shows the transferability of FADPC and the baseline methods. First, FADPC consistently surpasses other baseline in transferability, up to 15.4% better attack success rate. The average transferability of FADPC, AdvPC, 3D-Adv and KNN are 35.3%, 24.9%, 11.5%, and 8.92%, respectively. Overall, FADPC is superior to other methods in terms of transferability. When FADPC uses DGCNN as the target network to generate adversarial examples, the average transferability is 45.77%, the success rate is 28.17% when PointNet is the target network, and the success rate is 42.7% when PointNet++ (MSG) is the target network. When PointNet++ (SSG) is used as the target network, the success rate is 24.67%. In the transferability experiments of FADPC, we found that the adversarial examples generated by using DGCNN as the target network have the best universality, and the overall performance of the attack success rate is the highest.

4.2 Attack performance on defensive methods

In this section, we conduct the attack experiment on defensive methods and choose widely used SOR, SRS, DUP-Net [24] and confrontation training [18] methods. For evaluation, a drop rate of 10% is set for SRS. The parameters proposed in the original text are maintained in SOR. For DUP-Net, the ModelNet40 data set is trained with an up-sampling rate of 2. In the adversarial attack, all four networks are trained using a mixture of ModelNet40 training data and the generated adversarial examples.

Experimental results of attack success rates on defensive methods are shown in Table 2. The average attack success rate of FADPC on the four point cloud models with defense methods is 57.54%; the success rate of 3D-Adv, KNN and AdvPC are 33.65%, 35.65% and 53%, respectively. FADPC is more effective against SRS, a relatively simple statistical defense method, with an average attack success rate of 86.7%. It can be also found that DGCNN is the most difficult network to attack for all attack methods. Our method is to directly generate adversarial examples based on the original samples, thus the point distribution is closer to the original point cloud sample than other methods, which can help to escape from defense models. As a conclusion, the breakthrough defense performance of our method is the best on these four networks.

Table 2: Comparison of success rate of breaking through defense methods.

Model	Defense	3D-Adv	KNN	AdvPC	**FADPC**
PointNet	No defense	100	100	98.0	**100**
	ADV-training	9.2	10.0	43.6	**45.7**
	SOR	20.0	14.4	27.6	**35.5**
	DUP Net	12.0	9.2	15.6	**22.9**
	SRS	88.8	84.0	96.4	**96.8**
PointNet++ (MSG)	No defense	100	100	94.8	**98.4**
	ADV-training	18.8	46.0	48.4	**52.4**
	SOR	32.8	37.2	49.2	**55.3**
	DUP Net	31.6	33.6	42.8	**48.1**
	SRS	63.6	64.8	83.6	**86.9**
PointNet++ (SSG)	No defense	100	100	96.4	**98.7**
	ADV-training	20.8	19.2	74.4	**77.6**
	SOR	24.8	17.2	49.6	**55.1**
	DUP Net	18.4	15.2	33.6	**36.9**
	SRS	60.4	55.2	86.4	**88.2**
DGCNN	No defense	100	100	85.2	**94.7**
	ADV-training	16.8	37.6	48.0	**53.0**
	SOR	22.0	29.2	36.8	**46.6**
	DUP Net	34.8	36.0	36.8	**39.7**
	SRS	63.6	61.6	76.0	**79.9**

4.3 Speed and Resource Comparison

In order to evaluate the generation speed and resource consumption of our FADPC attack method, we test the speeds of FADPC and other batch generation methods (3D-Adv, AdvPC) to generate adversarial examples and the corresponding GPU resource consumptions. KNN is a single generation method, which takes a lot of time and resource occupancy, so no comparison is made. We ignore to use models with defensive strategies in this experiment.

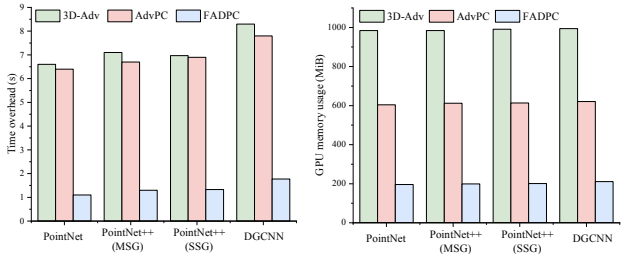


Figure 4: (a): Comparison of time consumption. (b): Comparison of GPU memory usage.

Figure 4(a) shows the generation time overheads of our method and other two baseline methods. All candidates take four kinds of networks as the target network to generate adversarial examples. The average time overheads for FADPC, AdvPC and 3D-Adv are 1.35 s, 6.9 s and 7.8 s, respectively. The generation speed of our method is 411% faster than that of AdvPC and 478% faster than that of 3D-Adv. Figure 4(b) shows the GPU memory consumed by our method, AdvPC and 3D-Adv. FADPC consumes 213 MiB GPU memory on average, AdvPC consumes 623 MiB, and 3D-Adv consumes 970 MiB. FADPC uses SOM to guide grouping, so that all points do not need to be calculated multiple times when extracting global features. This can speed up global feature extraction and reduce resource consumption. FADPC deploys two generation branches in its restoration network, which effectively avoids the shortcomings of only using a fully-connected network, leading to low time overhead and memory consumption. To sum up, the experimental results evaluate that FADPC has the fastest generation speed and the lowest resource occupation under the premise of ensuring the generation of high-quality adversarial examples.

5 CONCLUSION

In this paper, we have made efforts to generate adversarial examples for 3D point cloud deep learnings and proposed a self-organizing and parallel-process driven fast generation method. Specifically, SOM based feature extraction network is devised to extract global features and a dual-process based restoration network is constructed with a MLP branch and a convolution branch. With well-constructed hybrid loss function, these two sub-networks together can achieve fast but effective generation of adversarial examples for 3D point clouds. We also conducted benchmark-based experiments to evaluate the effectiveness of our method, which can achieve up to 511% faster speed with 34.19% lower resource usage while ensuring the success rates of attacks. Further more, our method can obtain better performance in the aspects of transferability and breaking through on defensive methods.

REFERENCES

[1] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. 2021. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *Proc. IEEE Symposium on Security and Privacy(SP)*. 176–194.

[2] Abdullah Hamdi, Sara Rojas, Ali Thabet, and Bernard Ghanem. 2020. AdvPC: Transferable adversarial perturbations on 3d point clouds. In *Proc. European Conference on Computer Vision(ECCV)*. 241–257.

[3] Wei Jiang, Zhiyuan He, Jinyu Zhan, Weijia Pan, and Deepak Adhikari. 2021. Research Progress and Challenges on Application-Driven Adversarial Examples: A Survey. *ACM Transactions on Cyber-Physical Systems (TCPS)* 5, 4 (2021), 1–25.

[4] Loic Landrieu and Mohamed Boussaha. 2019. Point cloud oversegmentation with graph-structured deep metric learning. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*. 7440–7449.

[5] Loic Landrieu and Martin Simonovsky. 2018. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*. 4558–4567.

[6] Jiaxin Li, Ben M Chen, and Gim Hee Lee. 2018. So-net: Self-organizing network for point cloud analysis. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*. 9397–9406.

[7] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. 2018. PointCNN: Convolution on x-transformed points. *Proc. Advances in neural information processing systems(NIPS)* (2018), 820–830.

[8] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[9] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. PointNet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*. 652–660.

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

[11] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. 2018. Tangent convolutions for dense prediction in 3d. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*. 3887–3896.

[12] Tzungyu Tsai, Kaichen Yang, Tsung-Yi Ho, and Yier Jin. 2020. Robust adversarial objects against deep learning models. In *Proc. AAAI Conference on Artificial Intelligence(AAAI)*. 954–962.

[13] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. 2020. Physically realizable adversarial examples for lidar object detection. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*. 13716–13725.

[14] Jiakai Wang, Aishan Liu, Zixin Yin, Shunchang Liu, Shiyu Tang, and Xianglong Liu. 2021. Dual Attention Suppression Attack: Generate Adversarial Camouflage in Physical World. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*.

[15] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. 2018. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*. 2569–2578.

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

[17] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*. 1912–1920.

[18] Chong Xiang, Charles R Qi, and Bo Li. 2019. Generating 3d adversarial point clouds. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*. 9136–9144.

[19] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. 2018. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610* (2018).

[20] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L Yuille. 2019. Improving transferability of adversarial examples with input diversity. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*. 2730–2739.

[21] Kaichen Yang, Xuan-Yi Lin, Yixin Sun, Tsung-Yi Ho, and Yier Jin. 2021. 3D-Adv: Black-Box Adversarial Attacks against Deep Learning Models through 3D Sensors. In *Proc. ACM/IEEE Design Automation Conference(DAC)*. 547–552.

[22] Kaichen Yang, Tzungyu Tsai, Honggang Yu, Max Panoff, Tsung-Yi Ho, and Yier Jin. 2021. Robust Roadside Physical Adversarial Attack Against Deep Learning in Lidar Perception Modules. In *Proc. ACM Asia Conference on Computer and Communications Security(AsiaCCS)*. 349–362.

[23] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. PU-NET: Point cloud upsampling network. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*. 2790–2799.

[24] Hang Zhou, Kejiang Chen, Weiming Zhang, Han Fang, Wenbo Zhou, and Nenghai Yu. 2019. DUP-NET: Denoiser and upsampler network for 3d adversarial point clouds defense. In *Proc. IEEE conference on computer vision and pattern recognition(CVPR)*. 1961–1970.