Abstract

3D point-cloud recognition with PointNet and its variants has received remarkable progress. A missing ingredient, however, is the ability to automatically evaluate point-wise importance w.r.t. classification performance, which is usually reflected by a saliency map. A saliency map is an important tool as it allows one to perform further processes on point-cloud data. In this paper, we propose a novel way of characterizing critical points and segments to build point-cloud saliency maps. Our method assigns each point a score reflecting its contribution to the model-recognition loss. The saliency map explicitly explains which points are the key for model recognition. Furthermore, aggregations of highly-scored points indicate important segments/subsets in a point-cloud. Our motivation for constructing a saliency map is by point dropping, which is a non-differentiable operator. To overcome this issue, we approximate point-dropping with a differentiable procedure of shifting points towards the cloud centroid. Consequently, each saliency score can be efficiently measured by the corresponding gradient of the loss w.r.t. the point under the spherical coordinates. Extensive evaluations on several state-of-the-art point-cloud recognition models, including PointNet, PointNet++ and DGCNN, demonstrate the veracity and generality of our proposed saliency map. Code for experiments is released on https://github.com/tianzheng4/PointCloud-Saliency-Maps.

1. Introduction

Point clouds, which comprise raw outputs of many 3D data acquisition devices such as radars and sonars, are an important 3D data representation for computer-vision applications [2][1][18][13][12]. Real applications such as object classification and segmentation usually require high-level processing of 3D point clouds [17][4][2][6]. Recent research has proposed to employ Deep Neural Network (DNN) for high-accuracy and high-level processing of point clouds, achieving remarkable success. Representative DNN models for point-cloud data classification include PointNet [10], PointNet++ [11] and DGCNN [20], which successfully handled the irregularity of point clouds and achieved high classification accuracy. Beyond that, a notable characteristic of PointNet and its variants is their robustness to furthest/random point dropping. [10] owes the robustness to the max pooling layer in PointNet, which only concentrates on a critical subset of a point cloud. In other words, the recognition result is mainly determined by those critical points such that dropping some other non-critical points does not change the prediction. We refer to the corresponding theory given in [10] as critical-subset theory. Despite identifying such an important subset, we observed that the critical-subset theory is too ambiguous, as it does not specify the importance of each point and subset. In this paper, we propose a simple method to construct a general saliency map for point-level and subset-level saliency assessment. Note in [14][15][9], saliency map is constructed for images to characterize the contribution of each pixel value to the recognition result. We extend this concept to point cloud, aiming at studying importance of each single point. Specifically, our method assigns a saliency score for each point, reflecting the contribution of a point to the corresponding model-prediction loss. A saliency map is important to better understand point-cloud data, in that: On the one hand, if one drops points with the highest saliency scores, model performance would decrease significantly, endowing the potential to build an adversarial attack model. On the other hand, if only points with the lowest scores are dropped, model performance would not change a lot. Somewhat surprisingly, we find dropping points with negative scores even leads to better recognition performance.

Despite simplicity in concept, how to construct such a
point-level saliency map is nontrivial. One possible solution is to drop all possible combinations of points and compute the loss changes after dropping those combinations, i.e., loss difference caused by those combinations. However, this simple brute-force method is impractical because the computational complexity scales exponentially w.r.t. the number of points in a point cloud. Instead, we propose an efficient and effective method to approximate saliency maps with a single backward step through DNN models. The basic idea is to approximate point dropping with a continuous point-shifting procedure, i.e., moving points towards the point-cloud center. This is intuitively valid because the point cloud center is supposed to be uninformative for classification. In this way, prediction-loss changes can be approximated by the gradient of the loss w.r.t. the point under a spherical coordinate system. Thus, every point in a point cloud is associated with a score proportional to the gradient of loss w.r.t. the point.

\[ F_\theta(x) \triangleq \{ F_{\theta,j} | j = 1...k \}, \]

whose input is a point cloud \( X \) and output is a probability vector \( F_\theta(X) \). The classification loss of the network is denoted as \( L(X, y; \theta) \), which is usually defined as the cross-entropy between \( F_\theta(X) \) and \( y \).

Point Contribution We define the contribution of a point/points in a point cloud as the difference between the prediction-losses of two point clouds including or excluding the point/points, respectively. Formally, given a point \( x_i \in X \), the contribution of \( x_i \) is defined as \( L(X', y; \theta) - L(X, y; \theta) \), where \( X' \triangleq \{ x_j : j = 1...N, j \neq i \} \). If this value is positive (or large), we consider the contribution of \( x_i \) to model prediction as positive (or large). Because in this case, if \( x_i \) is added back to \( X' \), the loss will be reduced, leading to more accurate classification. Otherwise, we consider the contribution of \( x_i \) to be negative (or small).

Image and Point-Cloud Saliency Map Existing works on model interpretation and vulnerability have constructed saliency maps for images to identify which pixels are critical to model-recognition and how those pixel values can influence the recognition performance \cite{14, 15, 9}. We first propose a similar saliency map for point cloud here. Point-cloud saliency map assigns each point \( x_i \) a saliency score, i.e., \( s_i \), to reflect the contribution of \( x_i \). Formally, the map can be denoted as a function \( S_\theta (\cdot) \) with input \( X \) and outputting a vector of length \( N \), i.e., \( \{ s_i \}_{i = 1...N} \). We expect higher (positive) \( s_i \) to indicate more (positive) contribution of \( x_i \). We can use point-dropping to verify the veracity of our saliency map.

Point Dropping Point dropping is a method to evaluate the veracity of our proposed saliency map. If our saliency map is accurate, then dropping points with the highest/positive/lowest/negative saliency scores will degrade/improve recognition performance. Ideally, high (positive) saliency scores indicate significant positive contributions to the recognition result. Thus after dropping points with the highest scores, we are expected to have \( \arg_j \max F_{\theta,j}(X') \neq y \), where \( X' \) is the remaining point cloud. On the contrary, especially when the dropped points have negative saliency scores, which means they contribute negatively to the prediction, we should have \( \arg_j \max F_{\theta,j}(X') = y \).

2. Preliminaries

2.1. Definition and Notations

Point Cloud A point cloud is represented as \( (X \triangleq \{ x_i \}_{i = 1...N}, y) \), where \( x_i \in \mathbb{R}^3 \) is a 3D point and \( N \) is the number of points in the point cloud; \( y \in \{ 1, 2, ..., k \} \) is the ground-truth label, where \( k \) is the number of classes. We denote the output of a point-cloud classification network as

\[ F_\theta(\cdot) \triangleq \{ F_{\theta,j} | j = 1...k \}, \]
PointNet and PointNet++
PointNet [10] applies a composition of single variable-functions, a max pooling layer, and a function of the max pooled features, which is invariant to point orders, to approximate the functions for point-cloud classification and segmentation. Formally, the composition can be denoted as \( \gamma \circ \text{MAX}\{ h(x_i) \} \), with \( h(\cdot) \) a single-variable function, \( \text{MAX} \) the max-pooling layer, and \( \gamma \) a function of the max pooled features (i.e., \( \text{MAX}\{ h(x_i) \} \)).

PointNet++ [11] is one extension, which applies PointNet recursively on a nested partitioning of the input point set, to capture hierarchical structures induced by the metric space where points live in. Compared to PointNet, PointNet++ is able to learn hierarchical features w.r.t. the Euclidean distance metric, and thus typically achieves better performance.

Dynamic Graph Convolutional Neural Network (DGCNN)
DGCNN [20] integrates a novel operation into PointNet, namely EdgeConv, to capture local geometric structures while maintaining network invariance to point-permutation. Specifically, the operation EdgeConv generates features that can describe the neighboring relationships by constructing a local neighborhood graph and applying convolutional-like operations on edges connecting neighboring pairs of points. EdgeConv helps DGCNN achieve further performance improvement, usually surpassing PointNet and PointNet++.

Critical-Subset Theory
For any point cloud \( X \), [10] proves that there exists a subset \( C \subseteq X \), namely critical subset, which determines all the max pooled features \( u \), and thus the output of PointNet. We briefly explain this theory in the following: a PointNet network can be expressed as \( F(X) = \gamma \circ u(X) \), where \( \gamma \) is a continuous function, and \( u(X) \) represents the max pooled features. Apparently, \( F(X) \) is determined by \( u(X) \). \( u(X) \) is computed by \( u(X) = \text{MAX}\{ h(x_i) \} \), where \( \text{MAX} \) (i.e., a special max-pooling layer) is an operator that takes \( N \) vectors as input and returns a new vector of the element-wise maximums. For the \( j \)th dimension of \( u \), there exists one \( x_i \in X \) such that \( u_j = h_j(x_i) \), where \( h_j \) is the \( j \)-th dimension of \( h \).

Aggregate all those \( x_i \) into a subset \( C \subseteq X \) such that \( C \) will determine \( u \) and thus \( \gamma \circ u \). [10] named \( C \) as critical subset. As we can see, this theory is applicable to network structures similar to \( \gamma \circ \text{MAX}\{ h(x_i) \} \), where a max-pooled feature is simply determined by one point, but not to networks with more complicated structures. Visually, \( C \) usually distributes evenly along the skeleton of \( X \). In this sense, for PointNet, the critical subset seems to include all the points critical to the recognition result. We refer the readers who are interested in more details to the appendix in [10]. Although the critical-subset theory helps to identify a salient point subset, we found that the theory does not specify point-level saliency yet, and it is also not an accurate and exhaustive way to characterize subset-level saliency.

3. Point-Cloud Saliency Map
In this section, we derive our proposed saliency map following the definitions in Section 2.1. Instead of dropping every point/subset and calculating the loss change (difference), we approximate point dropping by the procedure of shifting points to the spherical core (center) of a point cloud. Through this way, the nondifferentiable loss change caused by point-dropping can be approximated by differentiable loss change under a point-shifting operation, based on which a saliency map is constructed.

3.1. From Point Dropping to Point Shifting
Our idea is illustrated in Fig. 2. The intuition is that all the external (outward) points of a point cloud are supposed to determine the recognition result, because those points encode shape information of objects, while the points near the point center (inward) almost have no effect on the recognition performance. More concretely, outward corresponds to all original external points not shifted (to the center). Consequently, dropping a point has similar effects to shifting the point towards the center in terms of eliminating the effect of the point on the classification result. A more precise explanation for this intuition in theory is that the central points for all the point clouds are at the same position after coordinate translation so that their contribution to recognition can be neglected. Formally, we divide a point cloud into two parts \( \{ x_i^l \}, \{ c_i \} \), where \( \{ c_i \} \) represents the point subset at the centroid, and \( \{ x_i^l \} \) represents the remaining points on the surface. For a natural point cloud, \( \{ c_i \} \) is usually an empty set. The max-pooling layer in PointNet can be rewritten as \( \text{MAX}(h(x_i^l)) = \max\{ \text{MAX}(h(x_i)), \text{MAX}(h(c_i)) \} \), where \( \max(a, b) \) returns the element-wise maximum of \( a \) and \( b \). Since \( \{ c_i \} \) is the same for all the point clouds after coordinate transformation, determinant max-pooled features should mainly come from \( \text{MAX}(h(x_i^l)) \).

To verify our hypothesis, we conduct a proof-of-concept experiment: thousands of pairs of point clouds are generated by dropping 100/1024 points and shifting those 100/1024 points to the point cloud center respectively. Here we totally used three schemes to select those 100/1024 points, including furthest point-dropping, random point-dropping, and point-dropping based on our saliency map. We use PointNet for classification of both of the point clouds in every pair. For all those selection schemes, the

\*Median value of x, y, z coordinates
where \( \frac{\partial \delta}{\partial r} \) is by relative to the spherical core. Under this spherical coordinate \( \partial \) corresponding the orthogonal coordinates \( \varphi, \psi \) to the spherical core, \( \psi \) is invariant. In order to overcome this issue, we consider point shifting in the Spherical Coordinate System, where a point is represented as \( (r, \psi, \phi) \) with \( r \) distance of a point to the spherical core, \( \psi \) and \( \phi \) the two angles of a point relative to the spherical core. Under this spherical coordinate system, as shown in Fig. 2 shifting a point towards the center by \( \delta \) will increase the loss \( \mathcal{L} \) by \( -\frac{\partial \mathcal{L}}{\partial r} \delta \). Then based on the equivalence we established in section 3.1 we measure the contribution of a point by a real-valued score – negative gradient of the loss \( \mathcal{L} \) w.r.t. \( r \), i.e., \( -\frac{\partial \mathcal{L}}{\partial r} \). To calculate \( \frac{\partial \mathcal{L}}{\partial r} \) for certain point cloud, we use the medians of the axis values of all the points in the point cloud as the spherical core, denoted as \( x_c \), to build the spherical coordinate system for outlier-robustness \( \mathcal{L} \). Formally, \( x_c \) can be expressed as

\[ x_{cj} = \text{median}(\{x_{ij} \mid x_i \in X\}) \quad (j = 1, 2, 3), \]

(1)

where \( (x_{i1}, x_{i2}, x_{i3}) \) represent the axis values of point \( x_i \) corresponding the orthogonal coordinates \( (x, y, z) \). Consequently, \( \frac{\partial \mathcal{L}}{\partial r} \) can be computed by the gradients under the original orthogonal coordinates as:

\[ \frac{\partial \mathcal{L}}{\partial r_i} = \sum_{j=1}^{3} \frac{\partial \mathcal{L}}{\partial x_{ij}} x_{ij} - x_{cj} \frac{r_i}{r}, \]

(2)

where \( r_i = \sqrt{\sum_{j=1}^{3}(x_{ij} - x_{cj})^2} \). In practice, we apply a change-of-variable by \( \rho_i = r_i^{-\alpha} (\alpha > 0) \) to allow more flexibility in saliency-map construction, where \( \alpha \) is used to rescale the point clouds. The gradient of \( \mathcal{L} \) w.r.t. \( \rho_i \) can be calculated by

\[ \frac{\partial \mathcal{L}}{\partial \rho_i} = -\frac{1}{\alpha} \frac{\partial \mathcal{L}}{\partial r_i} r_i^{1+\alpha} \]

(3)

Define \( \delta_r \) as a differential step size along \( \rho \). Since \( \delta_r = -\alpha r^{-\alpha - 1} \delta_r \), shifting a point \( (r, \psi, \phi) \) by \( -\delta_r \) (i.e., \( \delta_r \) toward the center \( r = 0 \)) is equivalent to shifting the point by \( \delta_r \) if ignoring the positive factor \( \alpha r^{-\alpha - 1} \). Therefore, under the framework of \( \rho, \psi, \phi \), we approximate the loss change by \( \frac{\partial \mathcal{L}}{\partial r} \delta_r \), which is proportional to \( \frac{\partial \mathcal{L}}{\partial r} \). Thus in the rescaled coordinates, we measure the contribution of a point \( x_i \) by \( \frac{\partial \mathcal{L}}{\partial \rho_i} \), i.e., \( -\frac{1}{\alpha} \frac{\partial \mathcal{L}}{\partial r_i} r_i^{1+\alpha} \). Since \( \frac{1}{\alpha} \) is a constant, we simply employ

\[ s_i = -\frac{\partial \mathcal{L}}{\partial \rho_i} r_i^{1+\alpha} \]

(4)

as the saliency score of \( x_i \) in our saliency map. Note the additional parameter \( \alpha \) gives us extra flexibility for saliency-map construction, and optimal choice of \( \alpha \) would be problem specific. In the following experiments, we simply set \( \alpha \) to 1, which already achieves remarkable performance. For better understanding of our saliency maps, several maps are visualized in Fig. 3. We colorcode those points by the ranks of their saliency scores, i.e., larger number indicated higher saliency scores.

4. Point Dropping Algorithms

As stated in Section 2.1 point dropping can be used to verify the veracity of our saliency map. Therefore, we propose two point dropping algorithms in Section 4.1. For comparison with the critical-subset theory, we also tried several critical-subset based point dropping strategies, and present the most effective one in Section 4.2. For simplicity, we refer to dropping points with the highest scores as high-drop, dropping points with the lowest scores as low-drop, and the most effective critical-subset based strategy as critical in the followings. Except for verification, point dropping is also helpful for understanding subset-level (segment-level) saliency. For instance, after high-drop, the remaining fragmented point cloud will be recognized as another object, which means the dropped points belong to the most important segments in the object for recognition. Surprisingly, the points dropped by our saliency-map-based high-drop algorithms are always clustered as illustrated in Fig. 8 and the clusters are indeed the critical segments for object recognition even in human eyes.

4.1. Saliency-Map based Point Dropping

Based on the illustrations in Section 3.2 saliency maps are readily constructed by calculating gradients following \( \frac{\partial \mathcal{L}}{\partial r} \), which guide our point-dropping processes (algorithms). Algorithm 1 describes our iterative algorithm for point dropping. Note calculating saliency scores at once might be suboptimal because point dependencies have been ignored. To alleviate this issue, we propose to drop points iteratively.
such that point dependencies in the remaining point set will be considered when calculating saliency scores for the next iteration. Specifically, in each iteration, a new saliency map is constructed for the remaining points, and among them \( n/T \) points are dropped based on the current saliency map. In section 5.3, we set \( n/T = 5 \) for dropping points with the highest saliency scores and show that this setting is good enough in terms of improving the performance and understanding subset-level saliency with reasonable computational cost.

Algorithm 1 Iteratively drop points based on dynamic saliency maps

Require: Loss function \( \mathcal{L}(X, y; \theta) \); point cloud input \( X \), label \( y \), and model weights \( \theta \); hyper-parameter \( \alpha \); total number of points to drop \( n \); number of iterations \( T \).

for \( t = 0 \) to \( T \) do
    Compute the gradient \( g^t_i = \nabla_{x^t_i} \mathcal{L}(X^t, y; \theta) \)
    Compute the center by \( x^t_i \triangleq (x^t_{i1}, x^t_{i2}, x^t_{i3}) = \text{median}(x^t_{i1}, x^t_{i2}, x^t_{i3}) \)
    Compute \( r_i \frac{\partial \mathcal{L}}{\partial r_i} = (x^t_i - x^t_i) \cdot g^t_i \) (inner product)
    Construct the saliency map by \( s_i = -r_i^\alpha \frac{\partial \mathcal{L}}{\partial r_i} \)
    if high-drop then
        Drop the points with \( n/T \) lowest \( s_i \) from \( X^t \)
    else if low-drop then
        Drop the points with \( n/T \) highest \( s_i \) from \( X^t \)
    end if
end for
Output \( X^T \)

4.2. Critical-Subset based Point Dropping

To compare our saliency map with the critical-subset theory, we also propose several point-dropping strategies based on the critical-subset theory, e.g., randomly dropping points from the critical-subset one-time/iteratively and dropping the points that contribute to the most number of max-pooled features one-time/iteratively. Among all those critical-subset based schemes, dropping the points with contribution to the most number of max-pooled features (at least two features) iteratively provides the best performance. The strategy is illustrated in Algorithm 2. However, we found that even this scheme still performs worse than our saliency-map based point-dropping algorithm, which indicates that our saliency map is a more accurate measure on the point-level and subset-level saliency.

Algorithm 2 Iteratively drop points based on dynamic critical subset

Require: PointNet network \( f = \gamma \circ \text{MAX}\{h(x_i)\} \); point cloud input \( X \), label \( y \), and model weights \( \theta \); hyper-parameter \( \alpha \); total number of points to drop \( n \); number of iterations \( T \).

for \( t = 0 \) to \( T \) do
    Compute the indexes of points in the critical-subset (index list) is by \( \text{arg MAX}\{h(x_i)\} \)
    Count \( c_i \triangleq \) the frequency of \( i \) in the list (i.e., \( x_i \) determines \( c_i \) max-pooled features)
    Drop \( n/T \) points with the largest \( c_i \) from \( X^t \)
end for

Output \( X^T \)

5. Experiments

We verify our saliency map and point dropping algorithms by applying them to several benchmarks.

5.1. Datasets and Models

We use the two public datasets, 3D MNIST\( ^{[2]} \) and ModelNet40\( ^{[21]} \), to test our saliency map and point-dropping algorithms. 3D MNIST contains 6000 raw 3D point clouds generated from 2D MNIST images, among which 5000 are used for training and 1000 for testing. Each raw point cloud contains about 20000 3D points. To enrich the dataset, we

\[ \text{http://modelnet.cs.princeton.edu/} \]

\[ \text{https://www.kaggle.com/daavoo/3d-mnist/} \]

version/13
randomly select 1024 points from each raw point cloud for 10 times to create 10 point clouds, making a training set of size 50000 and a testing set of size 10000, with each point cloud consisting of 1024 points. ModelNet40 contains 12,311 meshed CAD models of 40 categories, where 9,843 models are used for training and 2,468 models are for testing. We use the same point-cloud data provided by [10], which are sampled from the surfaces of those CAD models. Finally, our approach is evaluated on state-of-the-art point cloud models introduced in section 2.2, i.e., PointNet, PointNet++ and DGCNN.

5.2. Implementation Details

Our implementation is based on the models and code provided by [10, 11, 20]. Default settings are used to train these models. To enable dynamic point-number input along the second dimension of the batch-input tensor, for all the three models, we substitute several Tensorflow ops with equivalent ops that support dynamic inputs. We also rewrite a dynamic batch-gather ops and its gradient ops for DGCNN by C++ and Cuda. For simplicity, we set the number of votes as 1. In all of the following cases, approximately 1% accuracy improvement can be obtained by more votes, e.g., 12 votes. Besides, incorporation of additional features like face normals will further improve the accuracy by nearly 1%. We did not consider these tricks in our experiments for simplicity.

5.3. Empirical Results

To verify the veracity of our saliency map, we compare our saliency-map-driven point dropping approaches with the random point-dropping baseline [10], denoted as rand-drop, and the critical-subset-based strategy introduced in Section 4.2, denoted as critical (only applicable to the PointNet structure). For simplicity, we refer to dropping n points with the lowest saliency scores as low-drop, and dropping n points with highest positive scores as high-drop in the followings. For low-drop, we found one iteration of Algorithm 1 is already enough to achieve a good performance. While for high-drop, as explained in 4.1 we set n/T = 5 when dropping points with the highest scores in order to achieve better performance. We will further explain why we use this setting in the parameter study.

Results on PointNet The performance of PointNet on 3D-MNIST test set is shown in Fig. 4. The overall accuracy of PointNet maintains 94% ~ 95% under rand-drop while varying the number of dropped points between 0 to 200. In contrast, high-drop reduces PointNet’s overall accuracy to 49.2%. Furthermore, it is interesting to see by dropping points with negative scores, the accuracy even increases compared to using original point clouds by nearly 1%. This is consistent for other models and datasets as shown below. For ModelNet40, as shown in Fig. 4, the overall accuracy of PointNet maintains 87% ~ 89% under rand-drop. However, our point-dropping algorithm can increase/reduce the accuracy to 91.4%/44.3%.

Results on PointNet++ The results for PointNet++ are shown in Fig. 5 which maintains 95% ~ 96% on 3D-MNIST under rand-drop, while our point-dropping algorithm can increase/reduce the accuracy to 97.2%/59.5%. On the ModelNet40 test set, PointNet++ maintains 88 ~ 91% under high-drop, which is better than the baseline 87% under rand-drop. However, our point-dropping algorithm can improve the accuracy to 91.4% under 10%.

*Aggregate classification scores from multiple rotations
90% overall accuracy under rand-drop, while our algorithm can increase/reduce the accuracy to 91.1%/58.5%.

Results on DGCNN The accuracies of DGCNN on 3D-MNIST and ModelNet40 test sets are shown in Fig. 6 respectively. Similarly, under rand-drop, DGCNN maintains 96% \( \sim \) 97% and 89% \( \sim \) 91% accuracies respectively. Given the same conditions, our algorithm is able to increase/reduce the accuracies to 97.2%/76.4% and 91.3%/64.2% respectively.

Visualization Several point clouds manipulated by high-drop are visualized in Fig. 8. For the point clouds shown in those figures, our saliency map and the iterative point-dropping algorithm successfully identify the important segments (i.e., the dropped segments) that distinguish them from other clouds, e.g., the base of the lamp. It is worth pointing out that human also recognize several point clouds in Fig. 8 as other objects. On the contrary, as shown in Fig. 9, low-drop is visually similar to a denoising process, i.e., dropping noisy/useless points scattered throughout point clouds. Although the DNN model misclassifies the original point clouds in some cases, dropping those noisy points could correct the model predictions.

Parameter Study We employ PointNet on ModelNet40 to study the impacts of the scaling factor \( \alpha \), the number of dropped points \( n \), and the number of iterations \( T \) to model performance. As shown in Fig. 7 \( \alpha = 1 \) is a good setting for Algorithm\(^\dagger\) since as \( \alpha \) increases, model prediction loss will slightly decrease. Besides, it is clear in Fig. 7(2nd) that our high-drop significantly outperforms rand-drop in terms of degrading model performance: the accuracy of PointNet still maintains over 80% under rand-drop with 600/1024 points dropped, while high-drop reduces the accuracy to nearly 0. In Fig. 7 (3rd), we show that more iterations lead to better performance. However, when it comes to low-drop, more iterations only slightly improve the performance but with more computational cost. Therefore, we recommend executing our algorithm for 20 iterations to identify the important subsets (high-drop), and for one iteration to denoise the point clouds (low-drop).

Generalization We also show the generalization performance of our algorithm in Fig. 7(4th). Specifically, we test the PointNet-generated subsets (after dropping high-score points) on the PointNet++ and DGCNN, and the accuracy still degrades a lot.

Discussion Among all the three state-of-the-art DNN models for 3D point clouds, DGCNN appears to be the most robust model to point dropping (missing), which indicates DGCNN depends more on the entire point cloud rather than certain point or segment. We conjecture the robustness comes from its structures designed to capture more local and global information, which is supposed to compensate for the information loss by dropping points or segments. On the contrary, PointNet does not capture local structures\(^\dagger\), making it the most sensitive model to point dropping.

6. Conclusion In this paper, a saliency-map is constructed for 3D point-clouds to measure the contribution (importance) of each point in a point cloud to model prediction loss. By approxi-
mating point dropping with a continuous point-shifting procedure, we show that the contribution of a point is approximately proportional to, and thus can be scored by, the gradient of loss w.r.t. the point under a scaled spherical-coordinate system. Using this saliency map, we further standardize the point-dropping process to verify the veracity of our saliency map on characterizing point-level and subset-level saliency. Extensive evaluations show that our saliency-map-driven point-dropping algorithm consistently outperforms other schemes such as the random point-dropping scheme and critical-subset based strategy, indicating that our saliency is a more accurate measure to quantify the point-level and subset-level saliency of a point cloud.
References


