PointCloud Saliency Maps

Tianhang Zheng Changyou Chen Junsong Yuan
State University of New York at Buffalo
{tzheng4, changyou, jsyuan}@buffalo.edu

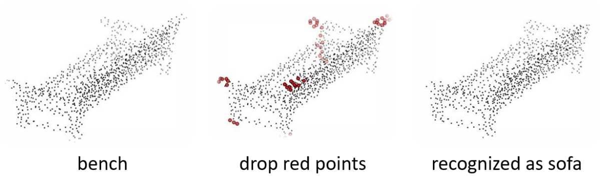
Bo Li Kui Ren
Zhejiang University
{boli, kuiren}@zju.edu.cn

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 [6, 17, 12, 11]. Real applications such as object classification and segmentation usually require high-level processing of 3D point clouds [16, 3, 1, 5]. 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 [9], PointNet++ [10] and DGCNN [19], 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. [9] 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 [9] 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 [13, 14, 8], 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.*



bench

drop red points

recognized as sofa

Figure 1. Drop the 5% most critical points identified by our saliency map from a bench point cloud can easily change the prediction outcome (*even can trick human vision!*).

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. *We further propose an iterative point-dropping algorithm for verification of our saliency map. As stated above, if our saliency map is effective, dropping points with the highest (positive)/lowest (negative) saliency scores will degrade/improve model performance.* Surprisingly, some point clouds manipulated by our point-dropping algorithm even concur with human intuition well as shown in Fig. 1, indicating our saliency map can recognize salient points and segments like human does.

We compared our saliency-map-driven point-dropping algorithms with the random point-dropping baseline and the best critical-subset-based strategy on several state-of-the-art point-cloud DNN models, including PointNet, PointNet++, and DGCNN. We show that our method can always outperform those schemes in terms of improving or degrading model performance with limited points dropped. As an example, we show that dropping $200/1024$ points with the highest saliency scores from each point cloud by our algorithm can reduce the classification accuracy of PointNet on 3D-MNIST/ModelNet40 to 49.2%/44.3%, while the random-dropping scheme only reduce the accuracy to 94.8%/87.7%, close to original accuracies. Besides, the best critical-subset-based strategy (*only applicable to PointNet*) only reduces the accuracies to 80.0%/58.1%. *All those experiments verified that our saliency map is a more accurate way to characterize point-level and even subset-level saliency than the critical-subset theory.*

2. Preliminaries

2.1. Definition and Notations

Point Cloud A point cloud is represented as $(\mathbf{X} \triangleq \{\mathbf{x}_i\}_{i=1...N}, y)$, where $\mathbf{x}_i \in \mathbb{R}^3$ 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 $\mathbf{F}_{\boldsymbol{\theta}}(\cdot) \triangleq \{F_{\boldsymbol{\theta},j} | j = 1...k\}$, whose input is a point cloud $\mathbf{X}$ and output is a probability vector $\mathbf{F}_{\boldsymbol{\theta}}(\mathbf{X})$. The classification loss of the network is denoted as $\mathcal{L}(\mathbf{X}, y; \boldsymbol{\theta})$, which is usually defined as the cross-entropy between $\mathbf{F}_{\boldsymbol{\theta}}(\mathbf{X})$ 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 $\mathbf{x}_i$ in $\mathbf{X}$, the contribution of $\mathbf{x}_i$ is defined as $\mathcal{L}(\mathbf{X}', y; \boldsymbol{\theta}) - \mathcal{L}(\mathbf{X}, y; \boldsymbol{\theta})$, where $\mathbf{X}' \triangleq \{\mathbf{x}_j : j = 1...N, j \neq i\}$. If this value is positive (or large), we consider the contribution of $\mathbf{x}_i$ to model prediction as positive (or large). Because in this case, if $\mathbf{x}_i$ is added back to $\mathbf{X}'$, the loss will be reduced, leading to more accurate classification. Otherwise, we consider the contribution of $\mathbf{x}_i$ 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 [13, 14, 8]. We first propose a similar saliency map for point cloud here. *Point-cloud saliency map assigns each point* $\mathbf{x}_i$ *a saliency score, i.e.,* s_i*, to reflect the contribution of* $\mathbf{x}_i$. Formally, the map can be denoted as a function $S_{\boldsymbol{\theta}}(\cdot)$ with input $\mathbf{X}$ 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 $\mathbf{x}_i$. 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 $\arg_j \max F_{\boldsymbol{\theta},j}(\mathbf{X}') \neq y$, where $\mathbf{X}'$ 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 $\arg_j \max F_{\boldsymbol{\theta},j}(\mathbf{X}') = y$.

2.2. 3D Point-Cloud Recognition Models

There are three mainstream approaches for 3D object recognition: volume-based [20, 7], multi-view-based [15, 18, 21, 4], and point-cloud-based [9, 10, 19] approaches, which rely on voxel, multi-view-image, and point-cloud representations of 3D objects, respectively. In this work, we focus on point-cloud-based models.

PointNet and PointNet++ PointNet [9] 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 $\gamma \circ \underset{\mathbf{x}_i \in \mathbf{X}}{\mathsf{MAX}}\{\mathbf{h}(\mathbf{x}_i)\}$, with $\mathbf{h}(\cdot)$ a single-variable function, MAX the max-pooling layer, and γ a function of the max pooled features (*i.e.,* $\mathsf{MAX}\{\mathbf{h}(\mathbf{x}_i)\}$). PointNet plays a significant role in the recent development of point-cloud high-level processing, serving as a baseline for many following point-cloud DNN models. PointNet++ [10] 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 [19] 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 $\mathbf{X}$, [9] proves that there exists a subset $\mathbf{C} \subseteq \mathbf{X}$, namely critical subset, which determines all the max pooled features $\mathbf{u}$, and thus the output of PointNet. We briefly explain this theory in the following: a PointNet network can be expressed as $\mathbf{F}(\mathbf{X}) \triangleq \gamma \circ \mathbf{u}(\mathbf{X})$, where γ is a continuous function, and $\mathbf{u}(\mathbf{X})$ represents the max pooled features. Apparently, $\mathbf{F}(\mathbf{X})$ is determined by $\mathbf{u}(\mathbf{X})$. $\mathbf{u}(\mathbf{X})$ is computed by $\mathbf{u}(\mathbf{X}) = \underset{\mathbf{x}_i \in \mathbf{X}}{\mathsf{MAX}}\{\mathbf{h}(\mathbf{x}_i)\}$, where 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 jth dimension of $\mathbf{u}$, there exists one $\mathbf{x}_i \in \mathbf{X}$ such that $\mathbf{u}_j = \mathbf{h}_j(\mathbf{x}_i)$, where $\mathbf{h}_j$ is the $j-th$ dimension of $\mathbf{h}$. Aggregate all those $\mathbf{x}_i$ into a subset $\mathbf{C} \subseteq \mathbf{X}$ such that $\mathbf{C}$ will determine $\mathbf{u}$ and thus $\gamma \circ \mathbf{u}$. [9] named $\mathbf{C}$ as critical subset. As we can see, this theory is applicable to network structures similar to $\gamma \circ \underset{\mathbf{x}_i \in \mathbf{X}}{\mathsf{MAX}}\{\mathbf{h}(\mathbf{x}_i)\}$, where a max-pooled feature is simply determined by one point, but not to networks with more complicated structures. Visually, $\mathbf{C}$ usually distributes evenly along the skeleton of $\mathbf{X}$. 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 [9].* 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 $\{\{\mathbf{x}'_i\}, \{\mathbf{c}_i\}\}$, where $\{\mathbf{c}_i\}$ represents the point subset at the centroid, and $\{\mathbf{x}'_i\}$ represents the remaining points on the surface. For a natural point cloud, $\{\mathbf{c}_i\}$ is usually an empty set. The max-pooling layer in PointNet can be rewritten as $\mathsf{MAX}\{\mathbf{h}(\mathbf{x}_i)\} = \max\{\mathsf{MAX}\{\mathbf{h}(\mathbf{x}'_i)\}, \mathsf{MAX}\{\mathbf{h}(\mathbf{c}_i)\}\}$, where $\max(\mathbf{a}, \mathbf{b})$ returns the element-wise maximum of $\mathbf{a}$ and $\mathbf{b}$. Since $\{\mathbf{c}_i\}$ is the same for all the point clouds after coordinate transformation, determinant max-pooled features should mainly come from $\mathsf{MAX}\{\mathbf{h}(\mathbf{x}'_i)\}$.

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

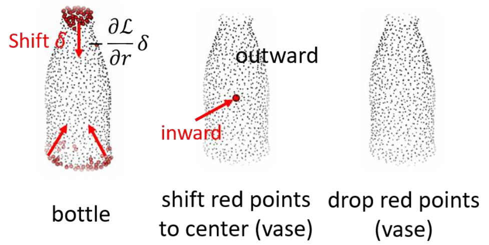

Shift δ
$-\frac{\partial \mathcal{L}}{\partial r}\delta$
outward
inward
bottle
shift red points to center (vase)
drop red points (vase)

Figure 2. Approximate point dropping with point shifting toward the point-cloud center.

classification results achieve more than 95% pairwise consistency[†], indicating applicability of our approach.

3.2. Gradient-based Saliency Map

Based on the intuition in 3.1, we approximate the contribution of a point by the gradient of loss, i.e., the difference between the prediction-losses of two point clouds including or excluding the point, under the point-shifting operation. Note that measuring gradients in the original coordinate system is problematic because points are not view (angle) invariant. In order to overcome this issue, we consider point shifting in the Spherical Coordinate System, where a point is represented as (r, ψ, ϕ) with r distance of a point to the spherical core, ψ and ϕ 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 δ 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 $\mathbf{x}_c$, to build the spherical coordinate system for outlier-robustness [2]. Formally, $\mathbf{x}_c$ can be expressed as

$$\mathbf{x}_{cj} = median(\{\mathbf{x}_{ij} \mid \mathbf{x}_i \in \mathbf{X}\}) \;\; (j = 1, 2, 3), \qquad (1)$$

where $(\mathbf{x}_{i1}, \mathbf{x}_{i2}, \mathbf{x}_{i3})$ represent the axis values of point $\mathbf{x}_i$ 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:

$$\frac{\partial \mathcal{L}}{\partial r_i} = \sum_{j=1}^{3} \frac{\partial \mathcal{L}}{\partial \mathbf{x}_{ij}} \frac{\mathbf{x}_{ij} - \mathbf{x}_{cj}}{r_i}, \qquad (2)$$

where $r_i = \sqrt{\sum_{j=1}^{3} (\mathbf{x}_{ij} - \mathbf{x}_{cj})^2}$. 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 α is used to rescale the point clouds. The gradient of $\mathcal{L}$ w.r.t. ρ_i can be calculated by

$$\frac{\partial \mathcal{L}}{\partial \rho_i} = -\frac{1}{\alpha} \frac{\partial \mathcal{L}}{\partial r_i} r_i^{1+\alpha}. \qquad (3)$$

Define δ_ρ / δ_r as a differential step size along ρ / r. Since $\delta_\rho = -\alpha r^{-(\alpha+1)} \delta_r$, shifting a point (r, ψ, ϕ) by $-\delta_r$ (*i.e.,* δ_r towards the center $r = 0$) is equivalent to shifting the point by δ_ρ if ignoring the positive factor $\alpha r^{-(\alpha+1)}$. Therefore, *under the framework of (ρ, ψ, ϕ), we approximate the loss change by $\frac{\partial \mathcal{L}}{\partial \rho}\delta_\rho$, which is proportional to $\frac{\partial \mathcal{L}}{\partial \rho}$*. Thus in the rescaled coordinates, we measure the contribution of a point $\mathbf{x}_i$ 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

$$s_i = -\frac{\partial \mathcal{L}}{\partial r_i} r_i^{1+\alpha} \qquad (4)$$

as the saliency score of $\mathbf{x}_i$ in our saliency map. Note the additional parameter α gives us extra flexibility for saliency-map construction, and optimal choice of α would be problem specific. *In the following experiments, we simply set α 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 (4), 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

[†] For more than 95% pairs, the classification results of the two point clouds in each pair are the same (may be correct or wrong)

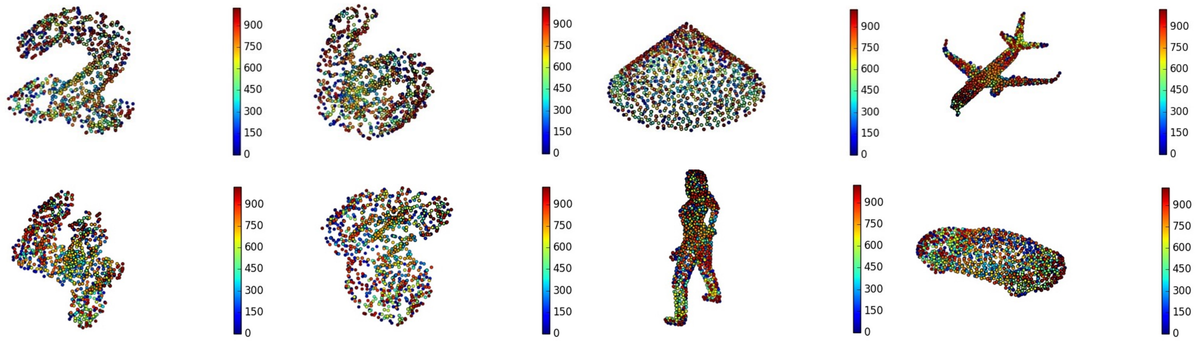


Figure 3. Visualize several saliency maps of digits and objectives (one-step): coloring points by their score-rankings.

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 $\mathcal{L}(\mathbf{X}, y; \boldsymbol{\theta})$; point cloud input $\mathbf{X}$,
  label $y$, and model weights $\boldsymbol{\theta}$; hyper-parameter $\alpha$; total
  number of points to drop $n$; number of iterations $T$.
  for $t$ = 0 to $T$ do
    Compute the gradient $\mathbf{g}_i^t = \nabla_{\mathbf{x}_i^t}\mathcal{L}(\mathbf{X}^t, y; \boldsymbol{\theta})$
    Compute the center by $\mathbf{x}_c^t \triangleq (x_{c1}^t, x_{c2}^t, x_{c3}^t) = median(\mathbf{x}_{i1}^t, \mathbf{x}_{i2}^t, \mathbf{x}_{i3}^t)$
    Compute $r_i \frac{\partial \mathcal{L}}{\partial r_i} = (\mathbf{x}_i^t - \mathbf{x}_c^t) \cdot \mathbf{g}_i^t$ (inner product)
    Construct the saliency map by $s_i = -r_i^{\alpha} r_i \frac{\partial \mathcal{L}}{\partial r_i}$
    if high-drop then
      Drop the points with $n/T$ lowest $s_i$ from $\mathbf{X}^t$
    else if low-drop then
      Drop the points with $n/T$ highest $s_i$ from $\mathbf{X}^t$
    end if
  end for
  Output $\mathbf{X}^T$
```

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 $f = \gamma \circ \underset{\mathbf{x}_i \in \mathbf{X}}{\mathsf{MAX}}\{h(\mathbf{x}_i)\}$; point
  cloud input $\mathbf{X}$, label $y$, and model weights $\boldsymbol{\theta}$; 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 $\arg \mathsf{MAX}\{h(\mathbf{x}_i)\}$
    Count $c_i \triangleq$ the frequency of $i$ in the list (i.e., $\mathbf{x}_i$ de-
    termines $c_i$ max-pooled features)
    Drop $n/T$ points with the largest $c_i$ from $\mathbf{X}^t$
  end for
  Output $\mathbf{X}^T$
```

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[‡] and ModelNet40[§] [20], 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

[‡] `https://www.kaggle.com/daavoo/3d-mnist/version/13`

[§] `http://modelnet.cs.princeton.edu/`

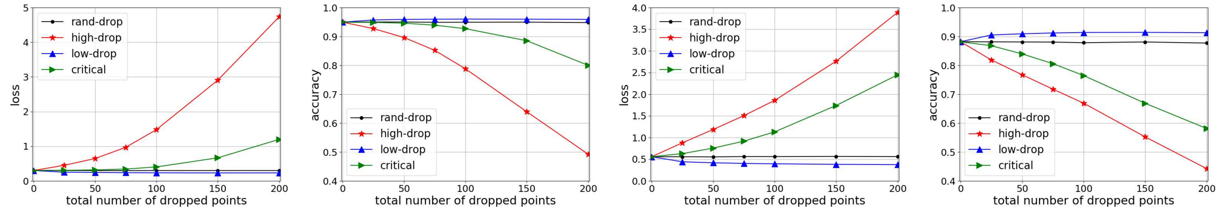


Figure 4. PointNet on 3D-MNIST and ModelNet40 from left to right: averaged loss (3D-MNIST), overall accuracy (3D-MNIST), averaged loss (ModelNet40), overall accuracy (ModelNet40).

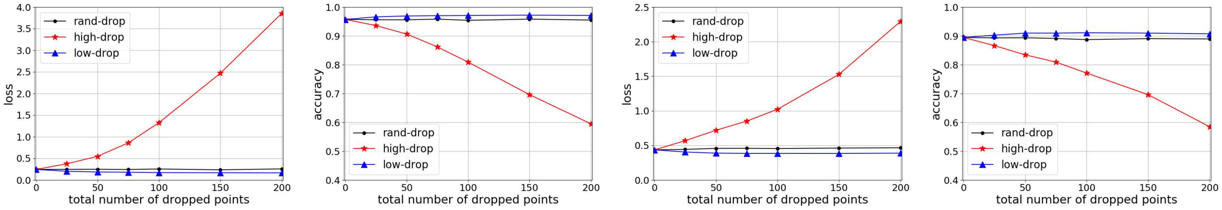


Figure 5. PointNet++ on 3D-MNIST and ModelNet40 from left to right: averaged loss (3D-MNIST), overall accuracy (3D-MNIST), averaged loss (ModelNet40), overall accuracy (ModelNet40).

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 [9], 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 [9, 10, 19] 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 [9], 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* $94\% \sim 95\%$ 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* $87\% \sim 89\%$ [*] 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 $95\% \sim 96\%$ 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 $88 \sim$

[¶] Aggregate classification scores from multiple rotations

[*] 89.2% in [9] can be acquired by setting the number of votes as 12. We set the number of votes to 1 for simplicity. The discrepancy between the accuracies under these two setting is always less than 1%.

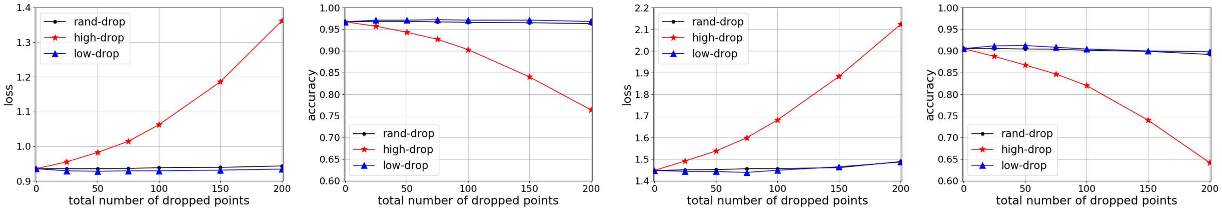


Figure 6. DGCNN on 3D-MNIST and ModelNet40: averaged loss (3D-MNIST), overall accuracy (3D-MNIST), averaged loss (ModelNet40), overall accuracy (ModelNet40).

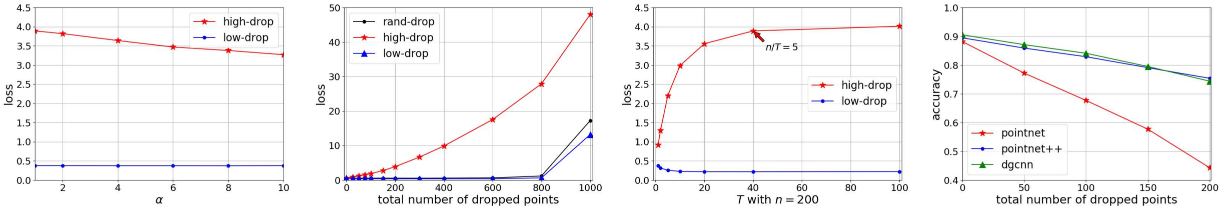


Figure 7. Impacts of hyper-parameters: (1) scaling factor α, (2) number of dropped points n (middle), (3) number of iterations T, (4) generalization results (subsets generated by point dropping on PointNet).

$90\%^{\dagger}$ 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 α, 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 1 since as α 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 [10], 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-

[†] 91.9% in [10] can be achieved by incorporating face normals as additional features and setting the number of votes as 12

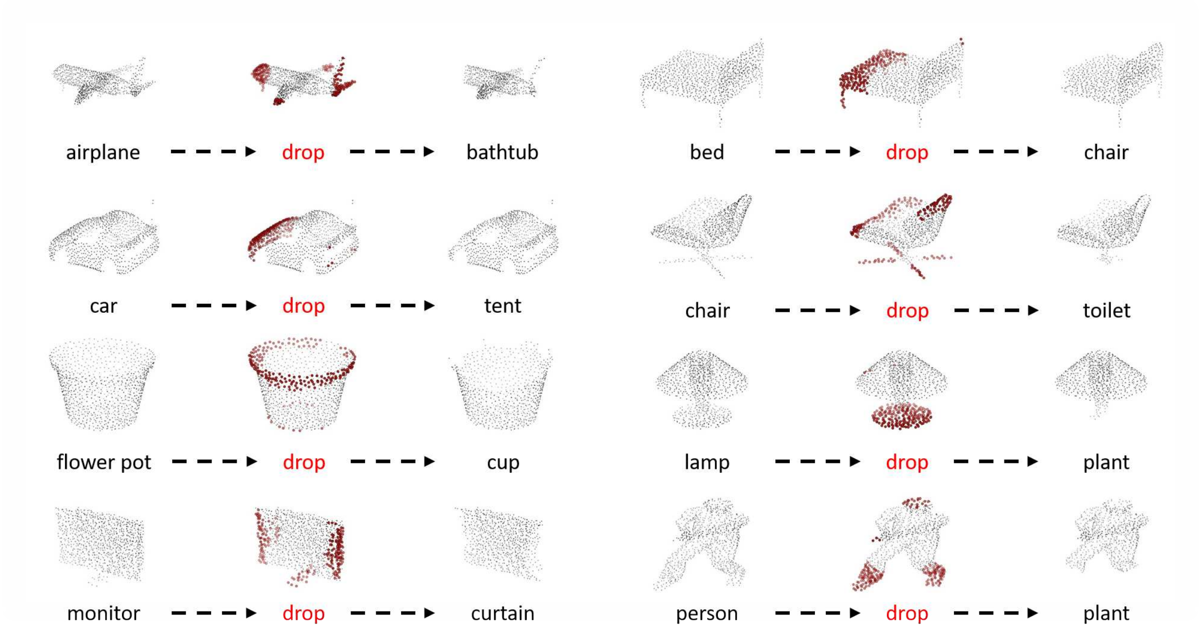


Figure 8. High-score point dropping (high-drop): original correct prediction (left), dropped points associated with *highest scores* by Algorithm 1 (middle), wrong prediction after point dropping (right).

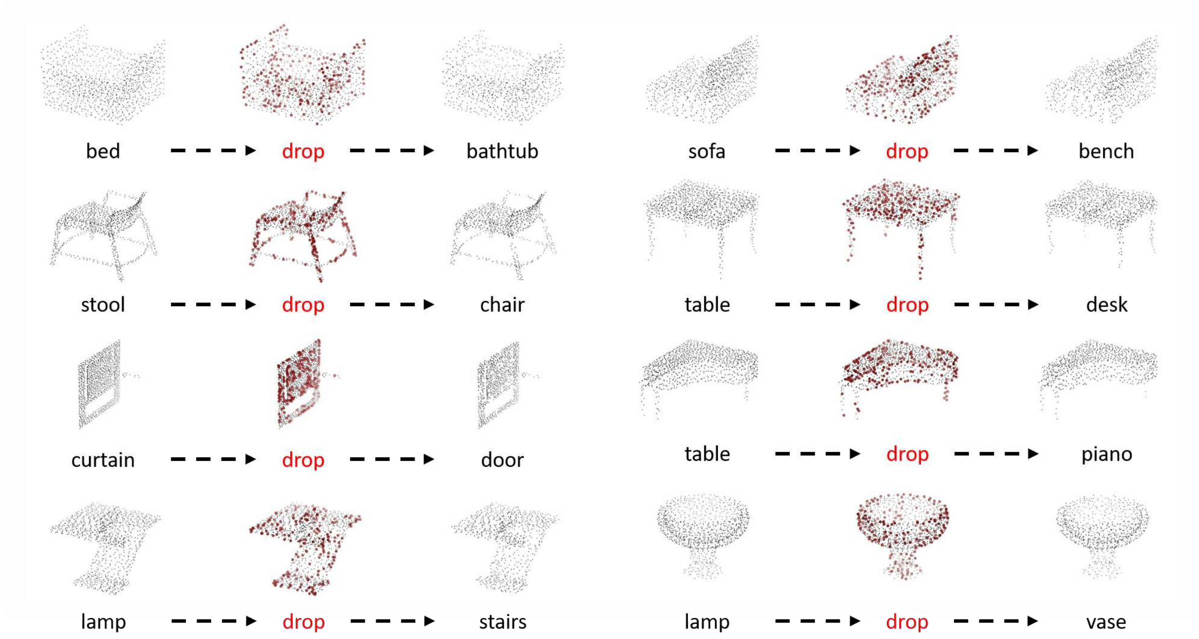


Figure 9. Low-score point dropping (low-drop): original wrong prediction (left), dropped points associated with *lowest* scores (middle), correct prediction after point dropping (right).

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

[1] Joydeep Biswas and Manuela Veloso. Depth camera based indoor mobile robot localization and navigation. In *2012 IEEE International Conference on Robotics and Automation*, pages 1697–1702. IEEE, 2012.

[2] Christian Böhm, Christos Faloutsos, and Claudia Plant. Outlier-robust clustering using independent components. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 185–198. ACM, 2008.

[3] Raia Hadsell, Pierre Sermanet, Jan Ben, Ayse Erkan, Marco Scoffier, Koray Kavukcuoglu, Urs Muller, and Yann LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.

[4] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[5] Ben Kehoe, Akihiro Matsukawa, Sal Candido, James Kuffner, and Ken Goldberg. Cloud-based robot grasping with the google object recognition engine. In *2013 IEEE International Conference on Robotics and Automation*, pages 4263–4270. IEEE, 2013.

[6] Lars Linsen. *Point cloud representation*. Univ., Fak. für Informatik, Bibliothek Technical Report, Faculty of Computer , 2001.

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

[8] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[9] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[10] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[11] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009.

[12] 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.

[13] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[14] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[15] 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.

[16] Sebastian Thrun, Michael Montemerlo, and Andrei Aron. Probabilistic terrain analysis for high-speed desert driving. In *Robotics: Science and Systems*, pages 16–19, 2006.

[17] George Vosselman, Ben GH Gorte, George Sithole, and Tahir Rabbani. Recognising structure in laser scanner point clouds. *International archives of photogrammetry, remote sensing and spatial information sciences*, 46(8):33–38, 2004.

[18] Chu Wang, Marcello Pelillo, and Kaleem Siddiqi. Dominant set clustering and pooling for multi-view 3d object recognition. In *Proceedings of British Machine Vision Conference (BMVC)*, volume 12, 2017.

[19] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[20] 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.

[21] 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.