Verifiability and Predictability: Interpreting Utilities of Network Architectures for Point Cloud Processing

Wen Shen²*, Zhihua Wei²*, Shikun Huang², Binbin Zhang², Panyue Chen², Ping Zhao², Quanshi Zhang¹†
¹Shanghai Jiao Tong University, Shanghai, China
²Tongji University, Shanghai, China
{wen.shen,zhihua.wei,hsk,0206zbb,2030793,zhaoping}@tongji.edu.cn,zqs1022@sjtu.edu.cn

Abstract

In this paper, we diagnose deep neural networks for 3D point cloud processing to explore utilities of different intermediate-layer network architectures. We propose a number of hypotheses on the effects of specific intermediate-layer network architectures on the representation capacity of DNNs. In order to prove the hypotheses, we design five metrics to diagnose various types of DNNs from the following perspectives, information discarding, information concentration, rotation robustness, adversarial robustness, and neighborhood inconsistency. We conduct comparative studies based on such metrics to verify the hypotheses. We further use the verified hypotheses to revise intermediate-layer architectures of existing DNNs and improve their utilities. Experiments demonstrate the effectiveness of our method. The code will be released when this paper is accepted.

1. Introduction

Recently, a series of works use deep neural networks (DNNs) for 3D point cloud processing and have achieved superior performance in various 3D tasks. However, traditional studies usually designed intermediate-layer architectures based on empiricism. Exploring and verifying utilities of each specific intermediate-layer architecture from the perspective of a DNN’s representation capacity still present significant challenges for state-of-the-art algorithms.

In this study, we aim to bridge the gap between the intermediate-layer architecture and its utilities. Table 1 lists three kinds of utilities considered in this study, including rotation robustness, adversarial robustness, and neighborhood inconsistency. Although there are many heuristic insights

<table>
<thead>
<tr>
<th>Intermediate-layer architectures</th>
<th>Rotation robustness</th>
<th>Adversarial robustness</th>
<th>Neighborhood consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Modules of using information of local density to reweight features [40].</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
</tr>
<tr>
<td>(b) Modules of using information of local coordinates to reweight features [40].</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>(c) Modules of concatenating multi-scale features [15].</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(d) Modules of computing orientation-aware features [11].</td>
<td>✓</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 1. Illustration of the verified utilities of specific intermediate-layer architectures. “–” denotes that the utility has not been examined, instead of indicating non-existence of the utility. Please see Fig. 1 for architectural details.

Experiments demonstrate the effectiveness of our method. The code will be released when this paper is accepted.

on utilities of existing architectures for 3D point cloud processing, there does not exist a rigorous and quantitative verification of such insights.

Therefore, we propose a method to quantitatively diagnose the utilities of intermediate-layer network architectures, which will provide new insights into architectural design. This is a necessary step towards the deep learning with scientific rigour. Note that, utilities are not necessarily equivariant to advantages. For example, in most cases, the rotation robustness is supposed to be a good property. However, the rotation robustness sometimes requires a DNN not to encode rotation-sensitive but discriminative features.

This study focuses on two terms, i.e. verifiability and predictability. In terms of verifiability, we design new metrics to quantify utilities of existing intermediate-layer architectures to prove intuitive insights. In terms of predictability, we further use the verified insights to revise other networks to improve their utilities. Note that, the revision of intermediate-layer architectures generally dose not change the depth of DNNs, so that we eliminate the influence of the depth change.

*Wen Shen and Zhihua Wei have equal contributions.
†Quanshi Zhang is the corresponding author. He is with the John Hopcroft Center and the MoE Key Lab of Artificial Intelligence, AI Institute, at the Shanghai Jiao Tong University, China.
More specifically, we propose a few hypotheses of utilities of specific intermediate-layer architectures, as shown in Table 1. Theoretically, we could analyze specific intermediate-layer architectures w.r.t. all utilities. However, due to the limit of the page number, we only verify hypotheses with strong connection to human intuitions. We design and conduct comparative studies to verify these hypotheses. The verified hypotheses are further used to guide the architectural revision of existing DNNs to improve their utilities. The verified hypotheses can be summarized as follows.

- The specific module in [40], which uses the local density information to reweight features (Fig. 1 (a)), improves the adversarial robustness (Table 1 (a)).
- Another specific module in [40], which uses local 3D coordinates’ information to reweight features (Fig. 1 (b)), improves the rotation robustness (Table 1 (b)).
- The specific module in [25, 18], which extracts multiscale features (Fig. 1 (c)), improves the adversarial robustness and the neighborhood consistency (Table 1 (c)). Neighborhood consistency measures whether a DNN assigns similar attention to neighboring points.
- The specific module in [11], which encodes the information of different orientations (Fig. 1 (d)), improves the rotation robustness (Table 1 (d)).

In order to verify the above hypotheses, we design the following five evaluation metrics and conduct a number of comparative experiments to quantify utilities of different intermediate-layer architectures.

1. Information discarding and 2. Information concentration: Information discarding measures how much information of an input point cloud is forgotten during the computation of a specific intermediate-layer feature. From the perspective of information propagation, the forward propagation through layers can be regarded as a hierarchical process of discarding input information [31]. Ideally, a DNN is supposed to discard information that is not related to the task. Let us take the task of object classification for example. The information of foreground points is usually supposed to be related to the task, while that of background points is not related to the task and is discarded.

To this end, we further propose information concentration to measure the gap between the information related to the task and the information not related to the task. Information concentration can be used to evaluate a DNN’s ability to focus on points related to the task.

3. Rotation robustness: Rotation robustness measures whether a DNN will use the same logic to recognize the same object when a point cloud has been rotated by a random angle. In other words, if two point clouds have the same global shape but different orientations, the DNN is supposed to select the same regions/points to compute the intermediate-layer feature. Unlike images with rich color information, point clouds usually only use spatial contexts for classification. Therefore, a well-trained DNN is supposed to have the rotation robustness.

4. Adversarial robustness: A reliable DNN is supposed to be robust to adversarial attacks.

5. Neighborhood inconsistency: Neighborhood inconsistency measures whether adjacent points have similar importance in the computation of an intermediate-layer feature. Adjacent points in a 3D object usually have similar shape contexts, so they are supposed to have similar importance. Therefore, ideally, a well-trained DNN should have a low value of neighborhood inconsistency.

The verified hypotheses are then applied to existing DNNs to revise their intermediate-layer architectures and improve their utilities. Note that this study aims to verify some insights about intermediate-layer architectures in the scenario of object classification, in order to improve utilities of existing DNNs. The classification accuracy is reported in supplementary materials.

Note that in comparative studies, unnecessarily complex intermediate-layer architectures usually bring in additional uncertainty, which will prevent our experiments from obtaining reliable and rigorous results. Therefore, we conduct experiments on simple-yet-classic intermediate-layer architectures.

Contributions of our study are summarized as follows.

(1) We propose a few hypotheses on utilities of specific intermediate-layer architectures. (2) We design five metrics to conduct comparative studies to verify these hypotheses, which provide new insights into architectural utilities. (3) It is proved that the verified hypotheses can be used to revise existing DNNs to improve their utilities.

2. Related work

Deep learning on 3D Point Cloud: Recently, many approaches use DNNs for 3D point cloud processing and have exhibited superior performance in various 3D tasks [24, 33, 36, 43, 42, 8, 37, 13, 30, 17, 29]. PointNet [24] was a pioneer in this direction, which used a max pooling layer to aggregate all individual point features into a global feature. However, such architecture fell short of capturing local features. PointNet++ [25] hierarchically used PointNet as a local descriptor to extract contextual information. Some studies [11, 39, 13, 19] further improved the networks’ ability to capture local geometric features. Some studies used graph convolutional neural networks for 3D point cloud processing [32, 39]. Others focused on the correlations between different regions of the 3D point cloud [18] or interaction between points [49]. In comparison, our study focuses on the utility analysis of intermediate-layer network architectures for point cloud processing.

1Values of the rotation robustness and the neighborhood consistency are negative numbers. For intuitive comparisons, we showed results of rotation non-robustness and neighborhood inconsistency in Section 5.
Visualization or diagnosis of representations: The most intuitive way to interpret DNNs is the visualization of visual patterns corresponding to a feature map or the network output [44, 22, 6, 51], such as gradient-based methods [7, 28], and the estimation of the saliency map [27, 20, 12, 24, 50]. In comparison, our study aims to explore the utility of intermediate-layer network architectures by diagnosing the information-processing logic of DNNs.

Quantitative evaluation of representations: In the field of explainable AI, explaining the capacity of representations has attracted increasing research attention. Some studies aimed to disentangle features of a DNN into quantifiable and interpretable feature components [48, 47, 46]. Some studies quantified the representation similarity to help understand the neural networks [9, 14, 23, 26]. Li et al. [15] quantitated the importance of different feature dimensions to guide model compression. Zhang et al. [45] quantitated the significance of interactions among multiple input variables of the DNN. Other studies explained the representation capacity of DNNs [4, 16, 38, 45]. The information-bottleneck theory [35, 31, 3] explained the trade-off between the information compression and the discrimination power of features in a neural network. Achille and Soatto [1] designed an information Dropout layer and quantified the information transmitted through it. Ma et al. [21] presented a method to calculate the entropy of the input information. Inspired by [21], we propose five metrics to diagnose feature representations of different DNNs and explore utilities of different intermediate-layer network architectures.

3. Metrics to Diagnose Networks

3.1. Preliminaries: quantification of entropy-based information discarding

We extend the method of calculating the entropy of the input information, which is proposed in [21], as the technical foundation. Based on this, a number of new metrics are designed to diagnose the DNN. The method quantifies the discarding of the input information during the layerwise forward propagation by computing the entropy of the input information given the specific feature of an intermediate layer. Given a point cloud $X$, let $f = h(X)$ denote the feature of a specific intermediate layer. It is assumed that $f'$ represents the same object concept$^2$ as $f$ when $f'$ satisfies $\|f' - f\|^2 < \epsilon$, where feature $f' = h(X')$, $X' = X + \delta$. $\delta$ denotes a random noise. Given a specific feature, the conditional entropy of the input information, when the input represents a specific object concept, i.e. we calculate entropy $H(X')$, s.t. $\|f' - f\|^2 < \epsilon$. It is assumed that $X'$ follows a Gaussian distribution $X' \sim N(X, \Sigma = diag(\sigma_1^2, \sigma_2^2, \ldots ))$. $\Sigma$ measures the maximum perturbation added to $X$ following the maximum-entropy principle, which subjects to $\|f' - f\|^2 < \epsilon$. Considering the assumption of the i.i.d. dimensions of $X'$, the overall entropy $H(X')$ can be decomposed into point-wise entropies.

$$
\max_{\sigma=[\sigma_1, \sigma_2, \ldots]} H(X'), \text{ s.t. } \|h(X') - f\|^2 < \epsilon^3,
$$

where $H(X') = \sum_i H_i$; $H_i = \log \sigma_i + \frac{1}{2} \log(2\pi\epsilon)$ denotes the entropy of the $i$-th point. $H_i$ quantifies how much information of the $i$-th point can be discarded, when the feature $h(X')$ is required to represent the concept of the target object.

3.2. Five metrics

Metric 1, information discarding: The information discarding is defined as $H(X')$ in Eqn. (1). The information discarding is measured at the point level, i.e. $H_i$, which quantifies how much information of the $i$-th point is discarded during the computation of an intermediate-layer feature. The point with a lower value of $H_i$ is regarded more important in the computation of the feature.

Metric 2, information concentration: The information concentration is based on the metric of information discarding. The information concentration is used to analyze a DNN’s ability to maintain the input information related to the task, and discard redundant information unrelated to the task. E.g., in the task of object classification, background points are usually supposed not to be related to the task and therefore more likely to be discarded by the DNN. Let $\Lambda^{\text{foreground}}$ denote the set of points in the foreground object in the point cloud $X$, and let $\Lambda^{\text{background}}$ denote the set of points in the background. Information concentration can be computed as the relative background information discarding w.r.t. foreground information discarding.

$$
\frac{\mathbb{E}_{\Lambda^{\text{background}}} [H_i] - \mathbb{E}_{\Lambda^{\text{foreground}}} [H_i]}{\mathbb{E}_{\Lambda^{\text{foreground}}} [H_i]},
$$

where a higher value of information concentration indicates that the DNN concentrates more on the foreground points during the computation of the feature.

Note that most widely used benchmark datasets for point cloud classification only contain foreground objects. Therefore, we generate a new dataset, where each point cloud contains both the foreground object and the background. In this new dataset, the background is composed of points that are irrelevant to the foreground. We will introduce details in Section 5.

Metric 3, rotation robustness: Given the two point clouds with the same shape but different orientations, the

$^2$In this study, the concept of an object is referred to as a small range of features that represent the same object instance.

$^3$We follow the [21] to slightly adjust the value of $\lambda$ to make the learned $\sigma_i$ satisfy $\mathbb{E}_{f'}[\|f' - f\|^2]$ is about twice the inherent variance of intermediate-layer features subject to a small input noise. $\lambda$ is a hyperparameter defined in [21].
rotation robustness is proposed to measure whether a DNN uses similar subsets of two point clouds to compute the intermediate-layer feature. Let $X_1 = \theta_1(X)$ and $X_2 = \theta_2(X)$ denote the point clouds that have the same shape but different orientations, where $\theta_1$ and $\theta_2$ denote two different rotation operations. To quantify the similarity of the attention on the two point clouds, we compute the Jensen-Shannon divergence between the distributions of information discarding in Eqn. (1), $\delta_1$ and $\delta_2$, w.r.t. $X_1 = \theta_1(X + \delta_1)$, $X_2 = \theta_2(X + \delta_2)$. I.e. we measure whether the DNN ignores similar sets of points to compute features of the two point clouds.

$$JSD(\delta_1||\delta_2), \quad \text{s.t.} \quad \begin{cases} \|h(X'_1) - h(X_1)\|^2 < \epsilon \\ \|h(X'_2) - h(X_2)\|^2 < \epsilon \end{cases}$$

where $\text{JSD}(\delta_1||\delta_2)$ measures the dissimilarity between information distributions over the two point clouds.

The rotation non-robustness$^3$ is defined as the average of the dissimilarity of attention on any two point clouds with different orientations, i.e. $\mathbb{E}_{\theta_1, \theta_2}[\text{JSD}(\delta_1||\delta_2)]$. In this study, we use the variational-approximation-based method in [10] to approximate the Jensen-Shannon divergence.

**Metric 4, adversarial robustness:** We use the method in [34] to perform adversarial attacks. The objective is

$$\min ||\epsilon||^2_2, \quad \text{s.t.} \quad C(X + \epsilon) = \hat{l} \neq l^*,$$

where $C(\cdot)$ is the predicted label; $l^*$ is the correct label of $X$; $\hat{l}$ is a target incorrect label. In this study, we perform targeted adversarial attacks against all incorrect classes. We use the average of $||\epsilon||_2$ over all incorrect classes to measure the adversarial robustness.

**Metric 5, neighborhood inconsistency$^1$:** The neighborhood inconsistency is proposed to evaluate a DNN’s ability to assign similar attention to neighboring points during the computation of an intermediate-layer feature. Ideally, for a DNN, except for special points (e.g. those on the edge), most neighboring points in a small region of a point cloud usually have similar shape contexts, so they are supposed to make similar contributions to the classification and receive similar attention, i.e. low neighborhood inconsistency. Let $N(i)$ denote a set of $K$ nearest points of the $i$-th point. We define the neighborhood inconsistency as the difference between the maximum and minimum point-wise information discarding within $N(i)$.

$$\mathbb{E}_i[\max_{j \in N(i)} H_j - \min_{j \in N(i)} H_j].$$

## 4. Hypotheses and Comparative Study

### 4.1. Overview of intermediate-layer architectures

- **Notation:** Let $x_i \in \mathbb{R}^3$ denote the $i$-th point, $i = 1, 2, \ldots, n$; let $N(i)$ denote a set of $K$ nearest points of $x_i$; let $F_i \in \mathbb{R}^{D \times K}$ denote intermediate-layer features of neighboring points $N(i)$, where each column of $F_i$ represents the feature of a specific point in $N(i)$.

- **Architecture 1, features reweighted by the information of the local density:** Architecture 1 focuses on the use of the local density information to reweight features [40]. As shown in Fig. 1 (a), for each point $x_i$, Architecture 1 uses the local density w.r.t. neighboring points of $x_i$ to compute $W^{H_1} \in \mathbb{R}^{K}$, which reweights intermediate-layer features $F_i$.

$$F'_i = F_i \, \text{diag}[W^{H_1}], \quad W^{H_1} = \text{mlp}(\text{density}(N(i))),$$

where $\text{diag}[W^{H_1}]$ transforms the vector $W^{H_1}$ into a diagonal matrix; $\text{density}(N(i))$ is a vector representing the density of neighboring points in $N(i)$; $\text{mlp}$ is a two-layer perceptron network.

- **Architecture 2, features reweighted by the information of local coordinates:** As shown in Fig. 1 (b), for each point $x_i$, Architecture 2 uses the information of local 3D coordinates to compute $W^{H_2} \in \mathbb{R}^{M \times K}$ to reweight intermediate-layer features $F_i$.

$$F'_i = F_i(W^{H_2})^\top, \quad W^{H_2} = \text{mlp}([x_j, j \in N(i)]),$$

where the $\text{mlp}$ is a single-layer perceptron network.

- **Architecture 3, multi-scale features:** Architecture 3 focuses on the use of multi-scale contextual information [25, 18]. As illustrated in Fig. 1 (c), $\{F_i^{\text{scale}=K_1}, \ldots, F_i^{\text{scale}=K_T}\}$ denote features that are extracted using contexts of $x_i$ at different scales, $F_i^{\text{scale}=K_i} \in \mathbb{R}^{D \times K_i}$. Each specific context w.r.t. $x_i$ is composed of $K_i$ nearest neighboring points around $x_i$. Then, $f_{\text{upper}}^{\text{scale}=K_1} \in \mathbb{R}^D$ in the upper layer is computed using $F_i^{\text{scale}=K_1}$. Architecture 3...
concatenates these multi-scale features to obtain $f_i^{upper}$.

$$f_i^{upper} = \text{concat} \left\{ f_i^{upper}, (\text{scale}=K_2) \right\} .$$

(8)

$$f_i^{upper} = \text{concat} \left\{ f_i^{upper}, (\text{scale}=K_3) \right\} .$$

(9)

where $g(\cdot)$ is a function for feature extraction. Details about this function are introduced in [24], which are also summarized in supplementary materials.

- **Architecture 4, orientation-aware features:** Architecture 4 focuses on the use of orientation information [11]. As illustrated in Fig. 1 (d), for each point $x_i$, $F_i^{oe} \in \mathbb{R}^{d \times O}$ denotes the feature of $x_i$, which encodes the information of various orientations, where $O$ is the number of orientations. Architecture 4 uses $F_i^{oe}$ to compute the orientation-aware feature $f_i^{oe} \in \mathbb{R}^{d}$.

$$f_i^{oe} = \text{Conv}^{oe}(F_i^{oe}),$$

(9)

where Conv$^{oe}$ is a special convolution operator. Details about this operator and the computation of $f_i^{oe}$ are introduced in [11].

### 4.2. Four hypotheses and comparative study design

**Hypothesis 1:** Architecture 1 designed by [40] (in Fig. 1 (a)) increases adversarial robustness.

This hypothesis is based on the observation that PointConv [40] has strong adversarial robustness, which may stem from Architecture 1. To verify this, we construct two versions of the PointConv for comparison, i.e., one with Architecture 1 and the other without Architecture 1.

To obtain the PointConv without Architecture 1, we remove all the modules of Architecture 1 from the original network (see the footnote), which are located behind the 2-nd, 5-th, 8-th, 11-th, and 14-th nonlinear transformation layers. The global architecture of PointConv is introduced in [40].

$$f_i^{upper} = \text{mlp}(F_i) \text{diag}[W^{H_1}] \implies f_i^{upper} = \text{mlp}(F_i),$$

(10)

where $f_i^{upper}$ is the feature in the upper layer; diag$[W^{H_1}]$ transforms the vector $W^{H_1}$ into a diagonal matrix.

**Hypothesis 2:** Architecture 2 designed by [40] (in Fig. 1 (b)) increases rotation robustness.

This hypothesis is proposed based on the observation that PointConv [40] has strong rotation robustness, which may stem from Architecture 2. To verify this, we construct two versions PointConv for comparison, i.e., one with Architecture 2 and the other without Architecture 2.

To obtain the PointConv without Architecture 2, we remove all the modules of Architecture 2, which are located before the 3-rd, 6-th, 9-th, 12-th, and 15-th nonlinear transformation layers. The global architecture of PointConv is introduced in [40].

$$f_i^{upper} = \text{mlp}(F_i)(W^{H_2})^T \implies f_i^{upper} = \text{mlp}(F_i).$$

(11)

**Hypothesis 3:** Architecture 3 used in [25, 18] (in Fig. 1 (c)) increases adversarial robustness and neighborhood consistency.

This hypothesis is inspired by [25, 18], which encodes multi-scale contextual information. To verify this hypothesis, we construct three versions of Point2Sequence for comparison. The baseline network of Point2Sequence concatenates features of 4 different scales to compute the feature in the upper layer, \{upper, $f_i^{upper}$, (scale=$K_1$), $f_i^{upper}$, (scale=$K_2$), $f_i^{upper}$, (scale=$K_3$), $f_i^{upper}$, (scale=$K_4$)\}. In this study, we set $K_1 = 128$, $K_2 = 64$, $K_3 = 32$, and $K_4 = 16$. The first network extracts features with three different scales, \{upper, $f_i^{upper}$, (scale=$K_1$), $f_i^{upper}$, (scale=$K_2$), $f_i^{upper}$, (scale=$K_3$)\}, and the second one extracts features with two different scales, \{upper, $f_i^{upper}$, (scale=$K_1$), $f_i^{upper}$, (scale=$K_2$)\}. The global architecture of Point2Sequence is introduced in [18].

**Hypothesis 4:** Architecture 4 designed by [11] (in Fig. 1 (d)) increases rotation robustness.

This hypothesis is proposed based on the observation that PointSIFT [11] exhibits strong rotation robustness. It may be because Architecture 4 ensures that features contain information from various orientations. To verify this hypothesis, we construct two versions of the PointSIFT for comparisons, i.e., one with Architecture 4 and the other without Architecture 4.

To get the PointSIFT without Architecture 4, we remove all modules of Architecture 4 from the original network (see the footnote), which are located before the 1-st, 3-rd, 5-th, and 7-th nonlinear transformation layers. The global architecture of PointSIFT is introduced in [11].
We apply our method to three benchmark DNNs, including PointNet++, Point2Sequence, and RSCNN. These DNNs were trained using three benchmark datasets, including the ModelNet40 where $W$ has been introduced in Eqn. (25).

We prove their rotation robustness.

Eqn. (13) shows how to add Architecture 2 behind a nonlinear transformation layer.

$$f_i^{upper} = mlp(F_i) \implies f_i^{upper} = mlp(F_i)(W^{H_2})^\top,$$

where $W^{H_2}$ denotes the formula of Architecture 2, which has been introduced in Eqn. (7).

Table 2 (b) shows details about how to obtain DNNs with Architecture 2. Global architectures of PointNet++, Point2Sequence, and RSCNN are introduced in [25]$^3$, [18]$^3$, and [19]$^3$.

Table 2 (c) shows details about how to obtain DNNs with Architecture 3. Architecture 3 used in [18] is added to PointNet++, Point2Sequence, and RSCNN in order to improve their adversarial robustness and neighborhood consistency.

Table 2 (d) shows details about how to obtain DNNs with Architecture 4. Architecture 4 designed by [11] is added to PointNet++, Point2Sequence, and RSCNN in order to improve their rotation robustness.

Table 2. Adding specific intermediate-layer architectures to existing DNNs to improve utilities.

4.3. Comparative study for the improvement of utilities of existing DNNs

In this section, we further prove that the verified four hypotheses can be used to revise existing intermediate-layer network architectures in order to improve their utilities. We apply our method to three benchmark DNNs, including PointNet++ [25], Point2Sequence [18], and RSCNN [19].

In Section 4.2, we remove specific intermediate-layer architectures from original DNNs. Actually, if we take the DNN without the specific intermediate-layer architecture as the original one (e.g. the PointConv without Architecture 1) and take the real original DNN as the revised one (e.g. the PointConv with Architecture 1), then it is naturally proved that the verified hypotheses can be used to revise DNNs to improve their utilities.

Nevertheless, in this section, we further prove that these specific intermediate-layer architectures can improve utilities of other DNNs.

Architecture 1 designed by [40] is added to PointNet++, Point2Sequence, and RSCNN in order to improve their adversarial robustness.

For each network, we construct two versions for comparison, i.e. one with Architecture 1 and the other without Architecture 1.

Table 2 (a) shows details about how to obtain DNNs with Architecture 1. Global architectures of PointNet++, Point2Sequence, and RSCNN are introduced in [25]$^3$, [18]$^3$, and [19]$^3$.

Eqn. (12) shows how to add Architecture 1 behind a nonlinear transformation layer.

$$f_i^{upper} = mlp(F_i) \implies f_i^{upper} = mlp(F_i) \ diag(W^{H_1}),$$

(12)

where $W^{H_1}$ denotes the formula of Architecture 1, which has been introduced in Eqn. (6).
As shown in Table 3, we measured the proposed metrics for the fully connected layer close to the network output\(^{10}\), which had 512 hidden units. We measured adversarial robustness by performing adversarial attacks over all incorrect classes. We found that PointNet++ and RSCNN had relatively higher values of information discarding. PointNet++ and RSCNN concentrated more on the foreground object. PointConv, DGCNN, PointSIFT, and RSCNN performed well in rotation robustness. PointConv and PointSIFT exhibited higher adversarial robustness. DGCNN and PointSIFT exhibited lower neighborhood inconsistency.

**Verifying hypotheses of utilities of specific network intermediate-layer architectures:** As shown in Table 4, the proposed four hypotheses had been verified. Architecture 1 improved the utility of adversarial robustness. One possible reason was that Architecture 1 considered distances between each point and its neighbors during the computing of densities, which increased the difficulty of adversarial attacks. Architecture 3 also improved the utility of adversarial robustness. We found that the utility of adversarial robustness increased with the scale number of features. The reason may be that concatenating features with different scales enhanced the representation capacity, so that it was more challenging to conduct adversarial attacks. Architecture 2 and Architecture 4 improved the utility of rotation robustness. The reason may be that both Architecture 2 and Architecture 4 extracted contextual information from coordinates of each point’s neighbors using non-linear transformations. Such contextual information improved rotation robustness. Besides, networks with Architecture 3 usually had lower neighborhood inconsistency than those without Architecture 3. DNNs that extracted features from contexts of more scales usually exhibited lower neighborhood inconsistency. One possible reason was that extracting multi-scale features enhanced connections between neighboring points.

**Improving utilities of existing DNNs:** In this experiment, we aimed to prove that utilities of existing DNNs could be improved by using the verified hypotheses to guide the architectural revision. To this end, we conducted comparative studies as designed in Table 2. As shown in Table 5, adding specific intermediate-layer architectures to existing DNNs improved their utilities. Specifically, adding modules of Architecture 1 improved the utility of adversarial robustness of PointNet++, Point2Sequence, and RSCNN. Adding modules of Architecture 2 improved the utility of rotation robustness of PointNet++, Point2Sequence, and RSCNN. Adding modules of Architecture 3 improved utilities of adversarial robustness and neighborhood consistency of both PointNet++ and RSCNN. Adding modules of Architecture 4 improved the utility of rotation robustness of PointNet++, Point2Sequence, and RSCNN.

**Relationship between utilities and accuracy:** Note that

---

\(^{8}\)The ShapeNet dataset for classification is converted from the ShapeNet part segmentation dataset.

\(^{9}\)Detailed discussions are presented in supplementary materials.

\(^{10}\)Theoretically, features of any layer can be used to analyze the representation capacity of DNNs.
Table 4. Verifying hypotheses of utilities of specific intermediate-layer network architectures. The column $\Delta$ denotes the difference of the utility of the network with the specific architecture w.r.t. the network without the specific architecture. $\Delta > 0$ indicates that the corresponding hypothesis has been verified. Experimental results show that the proposed four hypotheses were verified.

<table>
<thead>
<tr>
<th>Hypothesis</th>
<th>Action</th>
<th>Architecture</th>
<th>ModelNet40 dataset</th>
<th>ShapeNet dataset</th>
<th>3D MNIST dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypothesis 1</td>
<td>Keep/remove Architecture 1 from PointNet++ for adversarial robustness</td>
<td>4.878</td>
<td>5.076</td>
<td>0.249</td>
<td>2.407</td>
</tr>
<tr>
<td>Hypothesis 2</td>
<td>Keep/remove Architecture 2 from PointNet++ for rotation non-robustness</td>
<td>4.875</td>
<td>5.076</td>
<td>0.249</td>
<td>4.470</td>
</tr>
<tr>
<td>Hypothesis 3</td>
<td>Keep/remove Architecture 3 from Point2Sequence (4 vs. 3 scales) for adversarial robustness</td>
<td>2.526</td>
<td>2.521</td>
<td>0.005</td>
<td>2.520</td>
</tr>
<tr>
<td>Hypothesis 4</td>
<td>Keep/remove Architecture 4 from Point2Sequence for adversarial robustness</td>
<td>2.521</td>
<td>2.517</td>
<td>0.004</td>
<td>2.520</td>
</tr>
</tbody>
</table>

Table 5. Improving utilities of existing DNNs by adding modules of specific intermediate-layer architectures. The column “added” denotes the utility of the network which the specific architecture was added to. The column “ori.” denotes the utility of the original network. The column $\Delta$ denotes the improvement of the utility of the network which the specific architecture was added to w.r.t. the original network. In particular, for adversarial robustness, $\Delta$ was calculated as the difference of the utility of the network w/ the specific architecture minus the adversarial robustness of the network w/o the specific architecture. For rotation non-robustness and neighborhood inconsistency, $\Delta$ was calculated as the difference of the network non-robustness/neighborhood inconsistency of the network w/o the specific architecture minus the rotation non-robustness/neighborhood inconsistency of the network w/ the specific architecture.

<table>
<thead>
<tr>
<th>ModelNet40 dataset</th>
<th>ShapeNet dataset</th>
<th>3D MNIST dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Added Architecture 1 to PointNet++ for adversarial robustness</td>
<td>2.519</td>
<td>2.504</td>
</tr>
<tr>
<td>Added Architecture 1 to Point2Sequence for adversarial robustness</td>
<td>2.544</td>
<td>2.526</td>
</tr>
<tr>
<td>Added Architecture 1 to RSCNN for adversarial robustness</td>
<td>2.342</td>
<td>2.314</td>
</tr>
<tr>
<td>Added Architecture 2 to RSCNN for rotation non-robustness</td>
<td>3.993</td>
<td>3.645</td>
</tr>
<tr>
<td>Added Architecture 3 to PointNet++ for adversarial robustness</td>
<td>3.010</td>
<td>2.504</td>
</tr>
<tr>
<td>Added Architecture 3 to RSCNN for rotation non-robustness</td>
<td>2.350</td>
<td>2.314</td>
</tr>
<tr>
<td>Added Architecture 3 to RSCNN for neighborhood inconsistency</td>
<td>3.010</td>
<td>3.409</td>
</tr>
<tr>
<td>Added Architecture 3 to RSCNN for adversarial robustness</td>
<td>3.497</td>
<td>3.585</td>
</tr>
<tr>
<td>Added Architecture 4 to PointNet++ for rotation non-robustness</td>
<td>6.191</td>
<td>7.093</td>
</tr>
<tr>
<td>Added Architecture 4 to RSCNN for rotation non-robustness</td>
<td>2.424</td>
<td>3.645</td>
</tr>
</tbody>
</table>

this study focused on the verification of utilities of specific network architectures and the architectural revision of existing DNNs to improve their utilities, instead of the classification accuracy. Nevertheless, we provided the classification accuracy of different versions of DNNs in supplementary materials. Experimental results show that adding a specific architecture to existing DNNs has effects on accuracy. We have detailedly discussed the relationship between utilities and accuracy in supplementary materials.

6. Conclusion

In this paper, we have verified a few hypotheses of the utility of four specific intermediate-layer network architectures for 3D point cloud processing. Comparative studies are conducted to prove the utility of the specific architectures, including rotation robustness, adversarial robustness, and neighborhood inconsistency. In preliminary experiments, we have verified that Architecture 2 and Architecture 4 mainly improve the rotation robustness; Architecture 1 and Architecture 3 have positive effects on adversarial robustness; Architecture 3 usually alleviates the neighborhood inconsistency. These verified hypotheses have further been used to revise existing DNNs to improve their utilities.

Considering that unnecessarily complex intermediate-layer architectures will bring in uncertainty of experiments, we only verify utilities of simple network architectures w.r.t. the object classification. More generic hypotheses about utilities of other tasks (e.g., segmentation and reconstruction) need to be verified in the future.

**Acknowledgments** This work is partially supported by the National Nature Science Foundation of China (No. 61906120, U19B2043, 61976160).
References


[27] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any


