


SGG-Nets: Generic Rotation-Invariant Plugin Networks for Point Cloud Analysis

Jian Zhu , Jianrong Yan , Jiebin Huang , Yongwei Nie , *Member, IEEE*, Bin Sheng ,
and Tong-Yee Lee , *Senior Member, IEEE*

Abstract—Rotation invariance is a crucial requirement for the analysis of 3D point clouds. However, current methods often achieve rotation invariance by employing specific network designs. These networks, though perform well on rotation-aware tasks, is inferior in general tasks such as classification and segmentation. On the other hand, many powerful point processing networks, such as PointNet++, DGCNN, etc., have general point processing abilities, but do not own the property of rotation invariance. In this paper, we propose a standalone rotation-invariant convolution operator called SGGConv (Spherical Geometric Graph-based Convolution) and two ways integrating it with common point-based networks. The networks equipped with SGGConvs are called SGG-Nets which promote the rotation-invariance ability of regular point networks without modifying their network architectures much. Our contributions are three-fold. First, we propose a rotation-invariant feature descriptor, namely Spherical Geometry Descriptor (SGD), which captures point-pair features in a Local Spherical Coordinate System (LSCS). Second, we propose the SGGConv based on SGD and LSCS with an efficient Graph-based Spherical Feature Passing (GSFP) mechanism. Thirdly, we define two modules S-SGGConvMdl and M-SGGConvMdl, which are used to integrate SGGConv into baseline point nets. We test SGG-Nets, such as SGG-PointNet++, SGG-DGCNN, SGG-RICnnv++, on representative point cloud datasets. These models, equipped with our SGGConvs, not only enhance the rotation-invariance of

the baseline network but also improve its performance on point cloud analysis tasks such as classification and part segmentation, without incurring too much computational overhead.

Index Terms—Point cloud, rotation-invariance, spherical geometry descriptor, graph-based spherical feature passing.

I. INTRODUCTION

PPOINT clouds play an indispensable role in many fields of 3D vision, such as augmented reality, autonomous driving, and robotics. With the wide application of depth cameras and LiDAR, acquisition of point cloud data has become much easier and cheaper. It has become a hot research field to extract 3D features directly from point cloud through deep learning techniques for 3D point cloud analysis and processing [1], e.g., recognition [2], [3], [4], segmentation [5], registration [6], and completion [7], [8].

Existing deep 3D point networks can be roughly classified into two categories. The first category is referred to **regular networks** that are usually trained on benchmark datasets in which the point cloud data is usually pre-aligned and normalized, such as PointNet [2], PointCNN [9], DGCNN [10], PCT [11], Point Transformer [12], and PointVector [13]. These networks have general abilities in handling various 3D point cloud processing tasks. However, point cloud data acquired in real applications are usually not aligned due to different acquisition views or other factors. Therefore, there will be a sharp drop in performance when applying regular networks to the nonstandard data without alignment or normalization.

For the above reason, there is another category of methods referred to as **specialized networks** which are designed to learn robust representation of point cloud invariant to spatial transformations, e.g., translation and rotation [14]. Typically, they are either LRFs-based or PPFs-based. LRFs-based approaches, such as [15], [16], [17], [18], first construct Local Reference Frames (LRFs), and then convert the point coordinates from global to local frame, which are thus invariant to global rotations. PPFs-based approaches, e.g., [19], [20], [21], [22], try to first design rotation-invariant features, e.g., angles and distances, and then take these Point-Pair Features (PPFs) directly as the network input. In general, LRFs-based and PPFs-based methods have stronger ability for processing point clouds with unseen rotations. But they usually achieve rotation-invariance through specific designs. Although they outperform regular networks on metrics such as $SO3/SO3$ used to test rotation-invariance, they

Received 11 June 2024; revised 5 November 2024 and 10 December 2024; accepted 10 December 2024. Date of publication 17 February 2025; date of current version 27 August 2025. This work was supported in part by the National Natural Science Foundation of China (NSFC) for Key Program under Grant 62237001, in part by the NSFC for Excellent Young Scholars under Grant 62122022, in part by the NSFC for General Program under Grant 62072191, Grant 62206064, Grant 62206061, and Grant 62272298, in part by the Fundamental Research Funds for the Central Universities under Grant 2024ZYGXZR021, and in part by National Science and Technology Council under Grant 113-2221-E-006-161-MY3, Taiwan. The associate editor coordinating the review of this article and approving it for publication was Prof. Wei Tsang Ooi. (*Jian Zhu and Jianrong Yan contributed equally to this work.*) (*Corresponding authors: Yongwei Nie; Bin Sheng.*)

Jian Zhu is with the School of Computer Science and Technology, Guangdong University of Technology (GDUT), Guangzhou 510006, China (e-mail: dr.zhuj@gmail.com).

Jianrong Yan is with the School of Computer Science, Fudan University (FDU), Shanghai 200433, China (e-mail: jianrongyann@gmail.com).

Jiebin Huang and Yongwei Nie are with the School of Computer Science and Engineering, South China University of Technology (SCUT), Guangzhou 510006, China (e-mail: luckyjesbin@gmail.com; nieyongwei@scut.edu.cn).

Bin Sheng is with the Department of Computer Science and Engineering (CSE), Shanghai Jiao Tong University (SJTU), Shanghai 200240, China (e-mail: shengbin@cs.sjtu.edu.cn).

Tong-Yee Lee is with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 70101, Taiwan (e-mail: tonylee@mail.ncku.edu.tw).

This article has supplementary downloadable material available at <https://doi.org/10.1109/TMM.2025.3543001>, provided by the authors.

Digital Object Identifier 10.1109/TMM.2025.3543001

do not perform as well as regular networks on tasks such as classification and segmentation in many cases. In other words, regular networks and specialized networks each have their own advantages and disadvantages. The above analysis motivates us to combine the strengths of regular and specialized networks. Our goal is to enhance the ability of regular networks (e.g., PointNet++ [23], DGCNN [10], etc.) to handle objects with varying rotations. In this way, we can leverage the powerful capabilities of the well-established regular networks for general tasks, such as segmentation and classification, while addressing their limitations in processing rotated point clouds. To develop such methods, two requirements must be met: (1) The architectures of regular networks should remain largely unchanged; otherwise, they may become specialized networks. (2) The method should be easily applicable to various regular networks.

To achieve the above goals, we propose an effective convolution operator called Spherical Geometric Graph-based Convolution (SGGConv). Unlike previous specialized networks [15], [16], [17], [18], [19], [20], [21], [22], SGGConv is a general convolution module that can be conveniently integrated into common existing baseline point cloud networks to extract and transmit rotation-invariant features for the baseline networks, thereby improving their performance on both regular tasks and rotation-invariance tests. We call networks equipped with SGGConvs as **SGG-Nets**.

To establish SGGConv, we hand-craft a rotation-invariant feature descriptor named Spherical Geometry Descriptor (SGD), constructed with rotation-invariant point-pair features based on Local Spherical Coordinate System (LSCS), which have advantages of both LRFs-based method and PPFs-based method. Then, we construct a local neighborhood graph on the point cloud. Using SGDs as the features of the graph edges, we define an efficient Graph-based Spherical Feature Passing (GSFP) mechanism to update the node features (which are expected to be rotation-invariant) on the graph, forming SGGConv.

Then, to establish SGG-Nets, such as SGG-PointNet++ (combining SGGConvs with PointNet++) and SGG-DGCNN (combining SGGConvs with DGCNN), we define two convolutional modules based on SGGConvs, called S-SGGConvMdl and M-SGGConvMdl with single and multiple outputs respectively, where “Mdl” represents “module”. According to whether the baseline network contains subsampling operation (which will cause the reduction of the number of points), SGG-Nets select either S-SGGConvMdl or M-SGGConvMdl to establish a seamless connection between SGGConvs and the baseline network for integrating the learned rotation-invariant features. As a result, SGGConvs act as a plug-in to the baseline network for effective enhancement of the rotation-invariance without requiring too much change to the baseline.

We test the proposed SGG-Nets architecture on two representative point-based networks, including PointNet++ [23] and DGCNN [10]. Interestingly, SGGConvs can also be integrated with specialized networks such as RConv++ [22]. The models equipped with our SGGConvs outperform the state-of-the-arts in various aspects.

The main contributions of this article are summarized as follows.

- We propose a series of network architectures called SGG-Nets that are composed of regular networks and the proposed SGGConvs. SGG-Nets not only enhance the rotation-invariance of the existing regular networks, but also improve their performance in point cloud analysis tasks such as classification and segmentation.
- We construct a rotation-invariant feature descriptor called SGD based on Local Spherical Coordinate System (LSCS), which combines the advantages of both LRFs-based method and PPFs-based method.
- Based on SGD and LSCS, We define an efficient Graph-based Spherical Feature Passing (GSFP) mechanism to update the node features on a graph, which is the basis of our proposed SGGConv.
- We adaptively integrate SGGConvs into both regular and specialized networks with two types of convolutional modules (S-SGGConvMdl and M-SGGConvMdl). The improved networks outperform state-of-the-art methods in various tasks, showing their robustness, efficiency and generalizability.

II. RELATED WORKS

A. Deep Learning on Point Clouds

In 2017, PointNet [2] made a groundbreaking innovation by directly applying deep learning to point sets without any pre-processing. This pioneering work was further refined by PointNet++ [23] to overcome its limitations, laying the foundation for subsequent developments in deep networks. DGCNN [10] introduced EdgeConv, a technique that dynamically updates point features on graphs constructed layer by layer. Subsequent research efforts focused on various aspects of improving 3D point cloud learning. Some works [24], [25], [26] introduced self-attention networks for this purpose. Fast Point Transformer [24] proposed an architecture featuring a lightweight self-attention layer capable of effectively encoding the continuous 3D coordinates of large-scale point clouds. CFSA-Net [25] utilized Cross-Fusion Self-Attention (CFSA) to process large-scale point clouds efficiently, employing a hierarchical position encoding and residual optimization structure. CSAM [26] cyclically paired feature maps with position encoding to fully leverage spatial attention features and an adaptive fusion strategy. Despite these advancements, a significant challenge remained: many methods were highly sensitive to rotation, resulting in severe performance drops when dealing with arbitrarily rotated input point clouds.

B. Rotation-Invariant Networks for Point Clouds

Earlier methods (i.e., regular networks), such as PointNet [2] and PointNet++ [23], enhance the rotation-invariance of the network by means of data augmentation, such as rotation transformation, which usually leads to high spatial complexity and poor generalization ability to unseen rotations. Therefore, some specially-designed point cloud networks (i.e., specialized networks) are proposed for achieving rotation-invariance, and can

be roughly divided into two categories: Local Reference Frame (LRF)-based and Point Pair Feature (PPF)-based.

LRFs-based methods [15], [16], [17], [18] convert coordinates from global to local frames that are invariant to global rotation. L2G-GCN [15] proposes a Local-to-Global representation for point cloud to handle rotation transformation. PCA-RI [16] expresses point's coordinates in an intrinsic frame by the object shape itself to obtain rotation-invariant features. AECNN [17] proposes aligned edge CNNs to learn rotation-invariant features relative to LRFs. Zhao et al. [18] use a 3D capsule module to establish end-to-end transformation equivariance through a novel dynamic routing procedure on quaternions. Generally, LRFs-based methods can improve the performance of the network in tasks like classification and segmentation, but due to the limited relative position information captured, their generalization ability for unseen rotations is usually not very strong [27].

PPFs-based methods [19], [20], [21], [22] try to manually design rotation-invariant point-pair features and take them as the network input. SRINet [20] proposes the point projection feature and uses a novel architecture to extract features of different levels, thus ensuring rotation-invariance of the features. RICnv [21] uses low-level rotation-invariant geometric features, e.g., distances and angles, to design a convolution operator to achieve rotation-invariance of the network. Based on RICnv [21], RICnv++ [22] redesigns the rotation-invariant properties and encodes a wealth of them from different angles, thus further improving the performance. Note that PPFs-based methods can not only be used to improve classification accuracy but also for robust 3-D shape retrieval [28]. Compared with LRFs-based methods, PPFs-based methods usually have poorer performance in point cloud recognition tasks, but stronger generalization ability for rotation transformation.

A common drawback of both the LRFs-based and PPFs-based methods is that while they can perform well in tests for rotation-invariance, their performance in some common tasks such as classification and segmentation may not be superior to regular networks. Few methods exist that excel in both rotation-invariance tests and common tasks. Moreover, existing specialized networks often rely on complex designs, leading to relatively high computational costs.

C. 3D Graph Representation Learning

The representation of 3D graph data plays a crucial role in many applications such as 3D meshes, molecular biology and social networks. There have been various attempts in the literature to extend neural networks to deal with arbitrarily structured graphs. As a pioneer work, GCN [29] proposes to design fast localized convolutional filters on graphs for generalizing convolutional neural networks from regular grids to irregular graphs. Further, GAT [30] and GATv2 [31] take attention mechanism into account for more effective graph representation learning.

Shen et al. [32] define a learnable point-set kernel and use recursive feature aggregation on graph to take full advantage of a point's local neighborhood. PointWeb [33] uses an adaptive feature module to explore the interaction between the structural nodes of a point cloud graph. Shi et al. [34] propose an effective

graph neural network called Point-GNN to detect objects from a LiDAR point cloud. 3D-GCN [35] utilizes learnable kernels with a graph max-pooling mechanism to extract local 3D features from point clouds across scales. View-GCN [36] constructs view-graph from multiple views as node and designs a graph convolutional neural network over the view-graph for 3D shape analysis. Liu et al. [37] propose the spherical message passing (SMP) as a powerful feature aggregation scheme for representation learning of 3D chemical molecular graphs. He et al. [38] construct the local complete graph within each spherical voxel to learn the discriminative feature representation for 3D object detection. Recently, Du et al. [39] design two modules called local substructure encoding (LSE) and frame transition encoding (FTE) to improve the expressive power of 3D equivariant GNNs. In this paper, we represent the point cloud as a graph, assign rotation-invariant features to each edge in the graph, and propose an efficient convolution-like operation to update the node features.

III. APPROACH

A. Overview

This paper proposes a plug-in network structure called SGG-Net, designed to enhance the rotation invariance of existing point cloud processing networks. Our method consists of three main steps. First, we define a rotation-invariant geometric descriptor known as the Spherical Geometric Descriptor (SGD) based on a Local Spherical Coordinate System (LSCS), which combines the advantages of both LRF-based and PPF-based methods. Second, we construct a local neighborhood graph on the point cloud, and develop a Graph-based Spherical Feature Passing (GSFP) mechanism for effective rotation-invariant feature extraction. In this mechanism, SGD serves as the initial feature on the graph edges, and a generic rotation-invariant convolutional operator, SGGConv, is defined for feature updates on the graph. Third, depending on the baseline network architecture, we define two different convolutional modules, S-SGGConvMdl and M-SGGConvMdl, which serve as building blocks to effectively integrate SGGConvs into the baseline. The entire workflow is illustrated in Fig. 1.

In the following sections, we first introduce the definition of the rotation-invariant representation, SGD, in Section III-B, and then describe the proposed SGGConv and GSFP mechanism for feature updating on graph in Section III-C. Next, we detail the two convolutional modules, S-SGGConvMdl and M-SGGConvMdl, and explain how they can be integrated into the baseline network as plug-ins. Finally, the detailed network architectures of SGGConv-enhanced Networks (including SGG-PointNet++, SGG-DGCNN, SGG-RICnv++, etc.) are presented in Section III-E.

B. Rotation-Invariant Geometric Representation

Local Spherical Coordinate System (LSCS): The proposed SGGConv in this paper is a graph convolutional operator built on the raw point cloud. As SGGConv is integrated into baseline networks in order to enhance the rotation invariance of these

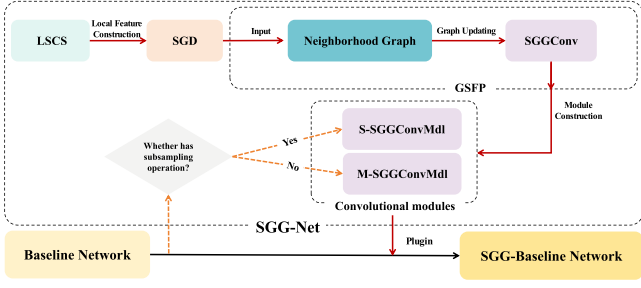


Fig. 1. The workflow of our proposed SGG-Net. Initially, a Local Spherical Coordinate System (LSCS) is defined for each point to construct a rotation-invariant geometric descriptor called the Spherical Geometric Descriptor (SGD). Subsequently, a neighborhood graph is constructed on the point cloud, and an efficient Graph-based Spherical Feature Passing (GSFP) mechanism is employed for updating the graph node features based on the defined convolution operation, SGGConv. Finally, depending on whether the baseline network includes subsampling operations, two specialized convolutional modules, S-SGGConvMdl and M-SGGConvMdl, are designed to integrate into the baseline network as plug-ins, thereby enhancing the network’s rotation invariance with minimal modifications to its architecture.

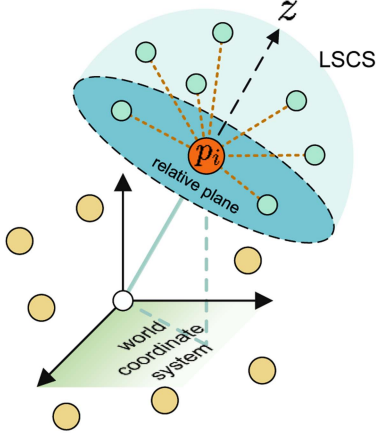


Fig. 2. Local Spherical Coordinate System (LSCS). For any reference point p_i , the LSCS is constructed by taking p_i as the origin, and the direction of \vec{p}_i as the z -axis. The plane perpendicular to z -axis and intersecting with p_i is defined as the Relative Plane.

networks, SGGConv should be insensitive to rotations of point clouds. Following LRFs-based and PPFs-based approaches, we consider using the relative positional information (e.g., relative distance, angle) of points as the input to SGGConvs, since they are rotation-invariant. Meanwhile, the input to the baseline network remains unchanged, consisting of 3D coordinates of the point cloud.

We extract information such as relative distance and angle in the Local Spherical Coordinate System (LSCS) as shown in Fig. 2. For any point p_i from a given point cloud P (i.e., $P = \{p_i | i = 1, 2, 3, \dots, N\} \in \mathbb{R}^{N \times 3}$, where N is the number of points in the cloud), we take it as the origin of the LSCS, and the direction of \vec{p}_i (which points from the origin of the world coordinate system to p_i) as the z -axis. Finally, we define the plane that is perpendicular to z -axis and intersects with p_i as the Relative Plane. The point p_i , the local z -axis, and the Relative Plane together form LSCS.

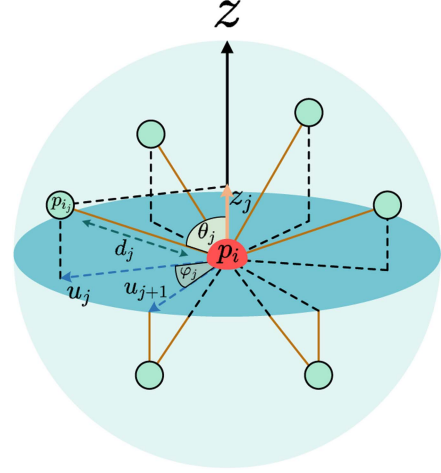


Fig. 3. Illustration of Spherical Geometric Descriptor (SGD). For each neighboring point p_{i_j} of p_i , after the LSCS is constructed, the SGD of p_{i_j} relative to p_i is defined as a 3-tuple $(d_j, \theta_j, \varphi_j)$, where d_j, θ_j, φ_j denote the radial distance, polar angle, and the torsion angle in the LSCS, respectively.

Spherical Geometric Descriptors (SGD): Now we have constructed the LSCS for point p_i . Suppose that p_{i_j} is the j -th neighbor of p_i , the relative location of p_{i_j} with respect to p_i can be specified by a 3-tuple $(d_j, \theta_j, \varphi_j)$, where d, θ, φ denote the radial distance, polar angle, and the torsion angle, respectively, as shown in Fig. 3. We define the tuple $(d_j, \theta_j, \varphi_j)$ as the Spherical Geometric Descriptor (SGD) of any neighboring point p_{i_j} with respect to the reference point p_i . Since the three variables are invariant to translations and rotations, we will establish SGGConv based on SGD later. Before that, we explain how to calculate d_j, θ_j and φ_j in detail at first.

First, the radial distance d_j between p_{i_j} and p_i can be obtained as:

$$d_j = \|p_{i_j} - p_i\|_2, \quad (1)$$

and the polar angle θ_j , which is the angle between $\vec{p_i p_{i_j}}$ and the z -axis, can be formulated as:

$$\theta_j = \arccos \frac{p_i \cdot (p_{i_j} - p_i)}{\|p_i\|_2 \cdot \|p_{i_j} - p_i\|_2 + \epsilon}, \quad (2)$$

where $(p_{i_j} - p_i)$ denotes the vector $\vec{p_i p_{i_j}}$, and ϵ is used to keep from being divided by zero and is set to 5×10^{-5} in our experiments.

Next, we define the projection vector of $\vec{p_i p_{i_j}}$ onto the z -axis and the Relative Plane as z_j and u_j , respectively.

$$z_j = \frac{(p_{i_j} - p_i) \cdot p_i}{\|p_i\|_2^2 + \epsilon} p_i, \quad (3)$$

$$u_j = (p_{i_j} - p_i) - z_j. \quad (4)$$

Then, the torsion angle φ_j which represents the angle between two adjacent pairs in the projection set $\{u_j | j = 1, 2, 3, \dots, k\}$ is formulated as:

$$\varphi_j = \arccos \frac{u_j \cdot u_{j+1}}{\|u_j\|_2 \cdot \|u_{j+1}\|_2 + \epsilon}, \quad (5)$$

where ± 1 denotes going clockwise or counterclockwise. In this way, any point p_{i_j} becomes the reference point for its next point $p_{i_{j+1}}$ when computing φ_j , and invariance is effectively achieved since reference point is naturally relative. Obviously, the sum of all the φ_j is 2π .

Similar to the LRFs-based methods, LSCS is established to capture the local geometric features and provides local context information. Besides, the construction of SGD skillfully combines the advantages of PPFs-based methods, because the relative position information of points is accurately obtained by defining point-pair features in the local spherical coordinate system of each point. In this way, we effectively integrate the advantages of LRFs-based and PPFs-based approaches.

C. Graph-Based Spherical Feature Passing (GSFP)

Graph Representation: To facilitate efficient learning on non-Euclidean point cloud data, we construct a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ on P , where $\mathcal{V} = \{p_1, \dots, p_N\}$ and \mathcal{E} is the set of edges constructed by connecting the K -nearest neighbors (K -NN) of each point. In particular, we use the SGD defined in Section III-B as the feature of the corresponding edge. Specifically, for any vertex p_i , the initial feature of the j -th edge e_j ($j \in \{1, \dots, K\}$) is thus given by:

$$e_j = [d_j, \theta_j, \varphi_j]. \quad (6)$$

In this way, we construct a graph with purely relative information as features. Subsequently, a convolution-like neural network will be built on this graph to learn useful feature information for enhancing the rotation-invariance of the baseline network.

SGGConv: We define a convolution-like operation called Spherical Geometry Graph Convolution (SGGConv) on the graph for efficient learning, by first updating the edge features with shared multi-layer perceptron (MLP) and then aggregating features from edges to nodes.

First, the update on any edge e_j can be described as:

$$e'_j = h_{\Theta}(e_j), \quad (7)$$

where $h_{\Theta} : \mathbb{R}^F \rightarrow \mathbb{R}^{F'}$ is a nonlinear function implemented by MLP with learnable parameters $\Theta = \{\phi_1, \dots, \phi_M\}$, and ϕ_m denotes the parameters of the m -th layer of the MLP. Specifically, the m -th layer of the update process can be formulated as:

$$e_j^{m+1} = \text{ReLU}(\phi_m \cdot e_j^m). \quad (8)$$

Note that we have $e_j^1 = e_j$ and $e_j^{M+1} = e'_j$, and we set $M = 2$ for efficiency.

Next, the SGGConv operation is accomplished by applying a channel-wise symmetric aggregation operation \square (Max or \sum) on the edge features to update the node features. The output of SGGConv at the i -th vertex is thus given by:

$$v_i = \square_{j \in \{1, 2, \dots, K\}} e'_j. \quad (9)$$

Like the convolutional operations in other graph neural networks such as EdgeConv in DGCNN [10], the aggregation operation \square in SGGConv is also set to Max. Intuitively, v_i contains rotation-invariant features of the nodes, because it integrates rotation invariance information from all associated edges.

TABLE I
DETAILED CONFIGURATION OF S-SGGCONVMDL AND M-SGGCONVMDL IN OUR IMPLEMENTATION

| Module | Layer | C_{in} | C_{out} | Input Size | Output Size |
|--------------|---------|----------|-----------|-------------------------|-------------------|
| S-SGGConvMdl | SGGConv | 3 | M_t | $N_t \times K \times 3$ | $N_t \times 6M_t$ |
| | SGGConv | M_t | M_t | | |
| | SGGConv | M_t | $2M_t$ | | |
| | SGGConv | $2M_t$ | $2M_t$ | | |
| M-SGGConvMdl | SGGConv | 3 | S | $N \times K \times 3$ | $N \times S$ |
| | SGGConv | S | S | | $N \times S$ |
| | SGGConv | S | $2S$ | | $N \times 2S$ |
| | SGGConv | $2S$ | $4S$ | | $N \times 4S$ |

D. Plugins: SGGConv-Based Convolutional Modules

In this section, we define two types of plugins composed of SGGConvs, which can be integrated with baseline networks (e.g., PointNet++ [23], DGCNN [10], etc.) to enhance their rotation-invariant feature extraction capabilities. The baseline networks considered in this paper can be divided into two categories based on whether they use subsampling in their network architecture. Subsampling is used to subsample the point cloud and reduce the scale of the input data. As the depth of the network increases, the dimension of the point features will also decrease. Therefore, subsampling the point cloud can effectively reduce computational complexity.

For baseline networks with subsampling operations (such as PointNet++[23]), the number of points changes. In this case, since different layers process point clouds with varying numbers of points, SGD can be computed multiple times to better adapt to these changes. For baseline networks without subsampling operations (such as DGCNN [10]), the point cloud remains static (meaning that the number of points does not change) throughout the network, so SGD is only computed once, which is sufficient for extracting rotation-invariant features from the static point cloud. Based on the above analysis, we define two convolutional modules: S-SGGConvMdl and M-SGGConvMdl.

S-SGGConvMdl: As shown in Fig. 4(a), the S-SGGConvMdl is a module specifically designed for baseline networks with subsampling operations. Specifically, we equip each subsampling layer in the baseline network with an S-SGGConvMdl to learn rotation-invariant features of the subsampled point cloud. Each subsampling layer provides a reduced number of points in the point cloud as input to the corresponding S-SGGConvMdl. An S-SGGConvMdl consists of multiple sequentially connected SGGConv layers, where the outputs from all SGGConv layers are concatenated and then fused with the point cloud features from the corresponding subsampling layer. In this case, we equip the baseline network with the same number of S-SGGConvMdl as the subsampling layers.

The detailed configuration of S-SGGConvMdl is shown in Table I, where C_{in}/C_{out} denotes the number of input/output channels, and N_t denotes the number of points involved in the t -th set subsampling layer. In our experiment, S-SSGGConvMdl

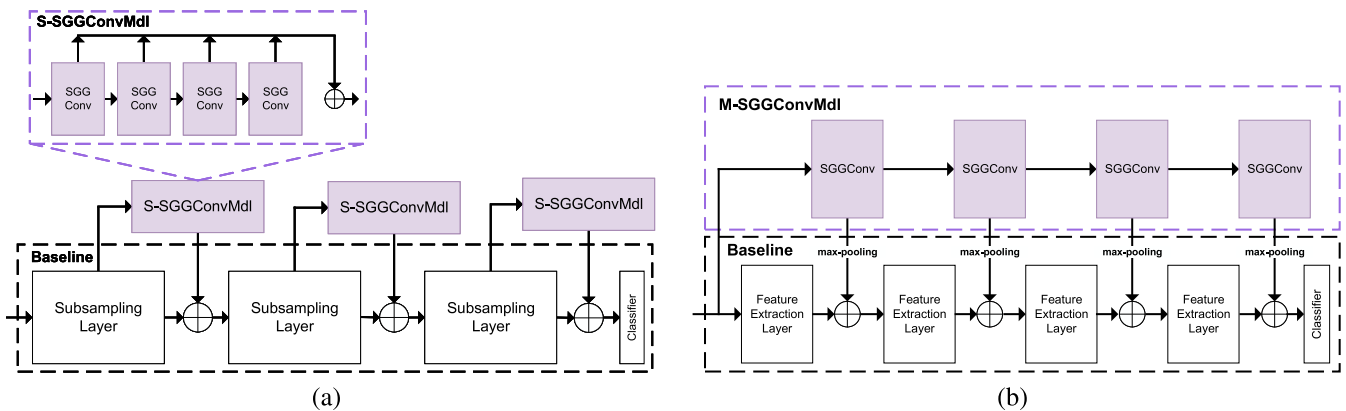


Fig. 4. Two different ways of combining SGGConv with the baseline network. (a) Integration with the baseline that includes subsampling layers. Each subsampling layer is equipped with an S-SGGConvMdl module (composed of several sequentially linked SGGConvs, where the outputs of each SGGConv are concatenated to form the module's output), and the concatenated output of both the SGGConvMdl module and the corresponding subsampling layer serves as the input to the next subsampling layer. (b) Integration with the baseline without subsampling layers. Depending on the number of feature extraction layers in the baseline, an M-SGGConvMdl module with the same number of SGGConvs is provided. Similar to S-SGGConvMdl, the SGGConvs in M-SGGConvMdl are also sequentially linked, but the output of each SGGConv is concatenated with the output of the corresponding feature extraction layer in the baseline and serves as the input to the next feature extraction layer. Note that \oplus denotes concatenation.

has four layers of SGGConv with M_t , M_t , $2M_t$, $2M_t$ output channels. M_t represents the number of SGGConv output channels, which is expected to be much smaller than the number of feature channels output by the corresponding subsampling block of the baseline network. The input of the first layer of SGGConv is SGD with a size of $N_t \times K \times 3$, and the input of each subsequent layer is the output of the previous layer, where N_t denotes the number of points after the t -th subsampling. S-SGGConvMdl concatenates the results of the four SGGConv layers as its output. After max pooling, the final output of $N_t \times 6M_t$ is obtained and passed to the subsampling block of the baseline network.

M-SGGConvMdl: As shown in Fig. 4(b), M-SGGConvMdl is a module specifically designed for baseline networks without subsampling operations, requiring only one M-SGGConvMdl per baseline network. Unlike S-SGGConvMdl, an M-SGGConvMdl generates multiple outputs. Specifically, both M-SGGConvMdl and the baseline network take the initial point cloud as input. The M-SGGConvMdl comprises multiple SGGConv layers, each producing a distinct feature output. Then the output is concatenated with the corresponding feature extraction layer in the baseline network.

As shown in Table I, M-SGGConvMdl has 4 layers of SGGConv with S , S , $2S$, $4S$ output channels in our experiment, where S denotes the number of SGGConv output channels at this layer. The input of the first layer of SGGConv is also SGD with size of $N \times K \times 3$, and the input of the latter layer is the output of the previous layer, where N represents the number of input point cloud (since there is no subsampling, the number of points is constant). Unlike S-SGGConvMdl, M-SGGConvMdl takes the result of each SGGConv layer as output and concatenates it to a feature extraction block of the baseline network after max pooling.

E. SGG-Nets: SGGConv-Enhanced Networks

In this section, we provide several examples of how to instantiate the aforementioned theory. Specifically, we integrate

SGGConvs with baseline networks using S-SGGConvMdl or M-SGGConvMdl to create SGG-Nets, i.e., SGGConv-enhanced networks. To demonstrate the network architecture of SGG-Nets without loss of generality, we first choose two regular networks, including PointNet++ [23] (which has subsampling operation) and DGCNN [10] (which does not have), as the baseline networks. In addition, to test whether SGG-Net can further improve the rotation-invariance of specialized networks (which already have dedicated designs to enhance rotation-invariance), we additionally choose RConv++ [22] and PaRot [40] as the baseline network. Note that although only four baseline networks are selected, we can integrate SGG-Net into other point cloud networks in a similar way.

In the following, we refer to the new networks obtained by integrating SGGConvs with the above four networks as SGG-PointNet++, SGG-DGCNN, SGG-RConv++, and SGG-PaRot, respectively. Due to space limitations, the detailed description of the SGG-PaRot network architecture is provided in the supplementary materials.

SGG-PointNet++: PointNet++ [23] employs a hierarchical structure, which is composed of a number of set abstraction levels to subsample the point set and capture local context at different scales. Fig. 5 shows the architecture of PointNet++ with two set abstraction levels, and each of them is integrated with an S-SGGConvMdl. For the first set abstraction level, SGD is computed based on the N_1 sampled points. The first S-SGGConvMdl takes SGD as input and outputs the corresponding rotation invariant feature with size of $(N_1, 6M_1)$, where M_1 is a hyper-parameter defined in Table I. Finally, the rotation invariant feature is concatenated with the output with size of $(N_1, d + C_1)$ from the first set abstraction level, which means we finish the first integration of S-SGGConvMdl with PointNet++. The result of concatenation is then used as the input to the second set abstraction level. After N_2 points are sampled from the input N_1 points in the second set abstraction level, we repeat the same process as done for the first set abstraction level.

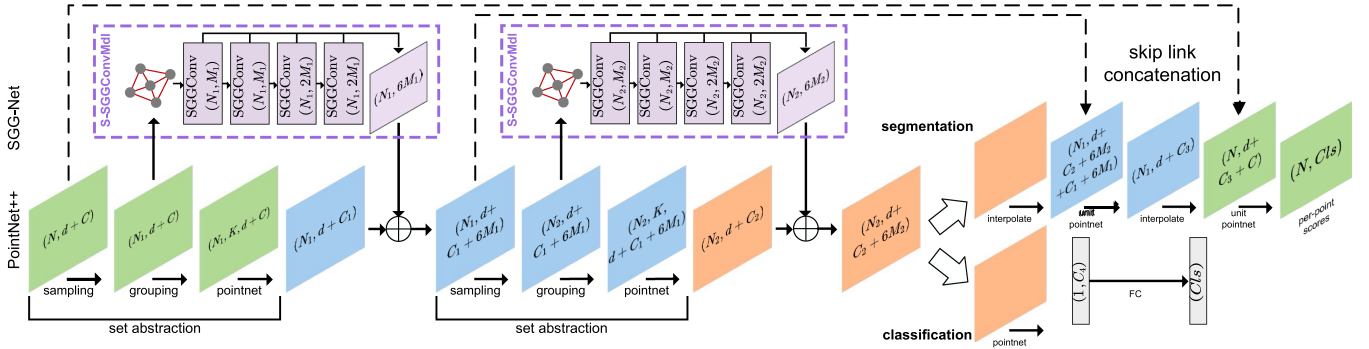


Fig. 5. Network architecture of SGG-PointNet++. For each set abstraction level (there are two), we first construct a neighborhood graph on the sampling points, and then use one S-SGGConvMdl module with four SGGConv layers to connect to the set abstraction level.

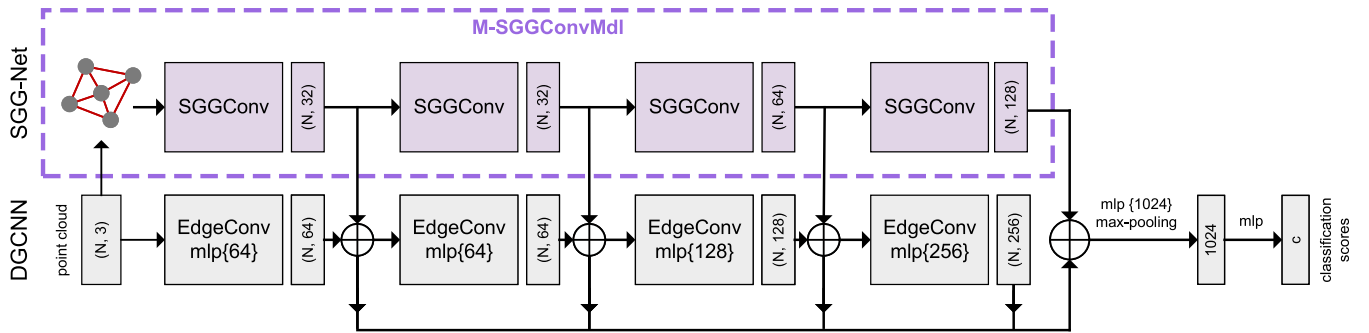


Fig. 6. Network architecture of SGG-DGCNN. We use just one M-SGGConvMdl module consisting of four SGGConv layers, and connect each of the SGGConvs to an EdgeConv layer of DGCNN.

In the implementation, PointNet++ [23] uses two set abstraction levels to subsample the input point cloud to $N_1 = 512$ and $N_2 = 128$ points, respectively. SGG-PointNet++ [23] uses two S-SGGConvMdl to extract rotation-invariant features. As shown in Fig. 5, PointNet++ [23] has a classification head and a segmentation head. For the classification head, the output feature dimensions (C_1 and C_2) of the two set abstraction levels in PointNet++ [23] are 320 and 640, while for the segmentation head they are 320 and 512. Accordingly, for the classification network, M_1 and M_2 are set to 32 and 64 respectively, which means that the output of the two S-SGGConvMdl is of size 512×192 (where 192 is computed as 32×6) and 128×384 (384 equals to 64×6), respectively. Thus, the results of concatenation, $(N_1, d + C_1 + 6M_1)$ and $(N_2, d + C_2 + 6M_2)$ in the first and the second set abstraction level are $(512, 3 + 320 + 192)$ and $(128, 3 + 640 + 384)$, respectively. In contrast, for the segmentation network, M_1 and M_2 are set to 32 and 48, respectively.

SGG-DGCNN: DGCNN [10] does not have point set subsampling operation, which implies that the number of points is fixed in the whole network, hence we only construct the neighborhood graph once for it, and use a single M-SGGConvMdl to adapt to the structure of DGCNN [10]. The network structure of SGG-DGCNN is shown in Fig. 6. We only need to simply concatenate the output of each SGGConv layer of M-SGGConvMdl with the corresponding output of the EdgeConv layer in DGCNN [10] and take it as the input to the next EdgeConv layer. As can

be seen, M-SGGConvMdl is introduced like a plugin without changing the structure of DGCNN [10].

Fig. 6 shows the classification network of DGCNN [10]. The segmentation network of DGCNN [10] has one less EdgeConv layer than the classification network. Accordingly, when designing the segmentation network of SGG-DGCNN, we use only the first three SGGConv layers in M-SGGConvMdl. Additionally, the value of S (see Table I) in M-SGGConvMdl is set to 32 for both classification and segmentation networks. This means that the dimension of the output feature of each SGGConv layer in M-SGGConvMdl is exactly half the dimension of the output feature of the corresponding EdgeConv layer.

SGG-RICov++: In RICov++ [22], the authors attempted to construct a rotation-invariant point cloud network by stacking multiple RICov++ layers which consists of rotation-invariant convolution operators. In our SGG-RICov++, the rotation-invariance is further improved. Similar to the set abstraction layer in PointNet++ [23], each RICov++ layer in RICov++ [22] contains a point set subsampling operation. Therefore, for each RICov++ layer, we connect one S-SGGConvMdl to it, and concatenate their outputs for the subsequent layer.

Taking the classification network of SGG-RICov++ as an example, its network architecture is shown in Table II. Specifically, RICov++ [22] includes five RICov++ layers, and the last layer only contains one point. Therefore, we equip S-SGGConvMdl modules for the first four layers, where the values of M_1 , M_2 ,

TABLE II
DETAILED CONFIGURATION FOR THE CLASSIFICATION NETWORK OF
SGG-RICONV++, WHERE THE BLUE PART IS INTRODUCED BY OUR SGG-NET

| Layer | Output Size |
|-----------------------|--------------------------|
| Input tensor | $3 \times N$ |
| RICONV++, S-SGConvMdl | $(32 + 24) \times 1024$ |
| RICONV++, S-SGConvMdl | $(64 + 36) \times 512$ |
| RICONV++, S-SGConvMdl | $(128 + 60) \times 256$ |
| RICONV++, S-SGConvMdl | $(256 + 120) \times 128$ |
| RICONV++ | 512×1 |
| Fully connected | 512×1 |
| Fully connected | 256×1 |
| Softmax | $K \times 1$ |

M_3 , and M_4 (see Table I) are set to 4, 6, 10, and 20, respectively. This means that the dimensions of the node features output by the four SSGConvMdl modules are 24, 36, 60, and 120, respectively. The segmentation network of SGG-RICONV++ is implemented in a similar way, whose network structure is not described in detail here.

IV. EXPERIMENTS

In this section, we conduct classification and segmentation tasks on ModelNet40 [41], ScanObjectNN [42], and ShapeNet [43]. More results on FG3D [44] and S3DIS [45] are provided in the supplementary materials. We compare SGG-Nets to their baseline versions and several representative point cloud networks to evaluate the effectiveness and efficiency of our approach. For the evaluation of rotation-invariance, we set up three training/testing settings: 1) training and testing are both performed with data under rotation around the z -axis (z/z), 2) training is performed with data under rotation around the z -axis while testing is performed with data under arbitrary rotation ($z/SO3$), and 3) training and testing are both performed with data under arbitrary rotation ($SO3/SO3$).

A. Classification on ModelNet40

Data: We first conduct classification tasks on ModelNet40 [41]. The dataset contains 12,311 CAD models with 40 categories, where 9,843 models are used for training and 2,468 models are used for testing. By default, we use 1024 points with 3D coordinate values (x, y, z) in Euclidean space as input to all the networks.

Implementation: For all the baseline networks and the selected networks for comparison, the training configuration follows the original papers. However, for fairness we train each model with a batch size of 32 for 300 epochs, and all the networks are executed on the same platform with two Tesla P100 GPUs.

Results: We perform experiments under two settings: regular training and testing (i.e., without any rotation transformation to the training and testing data), and training and testing with rotation transformations (including z/z , $z/SO3$, and $SO3/SO3$).

TABLE III
CLASSIFICATION RESULTS ON MODELNET40 WITH REGULAR TRAINING AND TESTING

| | Method | input | #points | OA (%) |
|----------------|----------------------------|---------|---------|---------------------|
| Specialized | RICONV++ [†] [22] | xyz | 1k | 90.6 |
| | PaRot [†] [40] | xyz | 1k | 90.5 |
| Regular | PointNet [2] | xyz | 1k | 89.2 |
| | PointNet++ [23] | xyz | 1k | 90.7 |
| | PointNet++ [23] | xyz+nor | 1k | 91.9 |
| | PointCNN [9] | xyz | 1k | 92.2 |
| | SO-Net [46] | xyz | 1k | 92.5 |
| | PAT [47] | xyz+nor | 1k | 91.7 |
| | PointConv [48] | xyz+nor | 1k | 92.5 |
| | DensePoint [49] | xyz | 1k | 92.8 |
| | RSCNN [50] | xyz | 1k | 91.7 |
| | KPConv [51] | xyz | 7k | 91.8 |
| | DGCNN [10] | xyz | 1k | 92.9 |
| | Point2Node [52] | xyz | 1k | 93.0 |
| | PointTransformer [12] | xyz | 1k | 93.7 |
| | PCT [11] | xyz | 1k | 93.2 |
| | PRA-Net [53] | xyz | 1k | 93.2 |
| EQ-Net [54] | xyz | 7k | 93.2 | |
| PointConT [55] | xyz | 1k | 93.5 | |
| Ours | SGG-PointNet++ | xyz | 1k | 92.6(↑ 1.9) |
| | SGG-PointNet++ | xyz+nor | 1k | 93.3(↑ 1.4) |
| | SGG-DGCNN | xyz | 1k | 93.7 (↑ 0.8) |
| | SGG-RICONV++ | xyz | 1k | 91.0(↑ 0.4) |
| | SGG-PaRot | xyz | 1k | 90.7 (↑ 0.2) |

The classification results are shown in Tables III and IV, respectively, where xyz denotes 3D coordinate values (x, y, z), and nor denotes surface normal vector. Besides, [†] denotes data from our experiments, otherwise from the original papers. Overall accuracy (OA) is used to evaluate the performance of the model.

As shown in Table III, the performance of specialized networks (see the ‘Specialized’ ones) designed specifically to enhance rotation-invariance is often not as good as regular networks (see the ‘regular’ ones) in the common classification tasks. After equipped with our SGG-Net, each baseline network can achieve a certain degree of improvement in the classification accuracy. Among the four baseline networks, SGGConvs have the greatest impact on improving PointNet++ [23], with an increase of 1.9% in the classification accuracy. Even for RICONV++ [22] and PaRot [40], which are both specifically designed to enhance rotation-invariance, the classification accuracy are improved by 0.4% and 0.2%, respectively. In addition, SGG-DGCNN has achieved the highest accuracy of 93.7% among all methods. All of these illustrate that SGGConvs can effectively improve the performance of baseline networks (including both regular and specialized ones) on regular classification tasks.

TABLE IV
CLASSIFICATION RESULTS ON MODELNET40 WITH ROTATION
TRANSFORMATION

| | Method | input | z/z | $z/SO3$ | $SO3/SO3$ |
|-------------|------------------------------|---------|---------------------|---------------------|---------------------|
| Specialized | SphericalCNN [56] | voxel | 88.9 | 76.9 | 86.9 |
| | SFCNN [57] | xyz | 91.4 | 84.4 | 90.1 |
| | RICnv [21] | xyz | 86.5 | 86.4 | 86.4 |
| | SRINet [20] | xyz | 87.0 | 87.0 | 87.0 |
| | ClusterNet [58] | xyz | 87.1 | 87.1 | 87.1 |
| | PCA-RI [16] | xyz | 88.8 | 88.8 | 88.8 |
| | L2G-GCN [15] | xyz | 89.5 | 89.5 | 89.5 |
| | SGMNet [27] | xyz | 90.0 | 90.0 | 90.0 |
| | LGANet [59] | xyz | 90.0 | 90.0 | 90.0 |
| | Li [60] | xyz | 89.4 | 89.4 | 89.3 |
| | RICnv++ [†] [22] | xyz | 90.8 | 91.0 | 90.9 |
| | PaRot [†] [40] | xyz | 90.6 | 90.6 | 90.9 |
| | Yu [61] | xyz | 91.0 | 91.0 | 91.0 |
| Regular | VoxNet [62] | voxel | 83.0 | - | 87.3 |
| | MVCNN 80x [63] | view | 90.2 | 81.5 | 86.0 |
| | SubVolSup [64] | voxel | 88.5 | 36.6 | 82.7 |
| | PointNet [2] | xyz | 87.0 | 21.6 | 80.3 |
| | PointNet++ [†] [23] | xyz | 89.3 | 31.5 | 88.5 |
| | PointNet++ [†] [23] | xyz+nor | 90.7 | 33.8 | 90.0 |
| | PointCNN [9] | xyz | 91.3 | 41.2 | 84.5 |
| | RS-CNN [50] | xyz | 90.3 | 48.7 | 82.6 |
| | DGCNN [†] [10] | xyz | 90.6 | 35.8 | 89.2 |
| Ours | SGG-PointNet++ | xyz | 90.8 (↑ 1.5) | 38.2 (↑ 6.7) | 89.7 (↑ 1.2) |
| | SGG-PointNet++ | xyz+nor | 91.4 (↑ 0.7) | 41.3 (↑ 7.5) | 90.6 (↑ 0.6) |
| | SGG-DGCNN | xyz | 91.2 (↑ 0.6) | 41.6 (↑ 5.8) | 90.8 (↑ 1.6) |
| | SGG-RICnv++ | xyz | 91.2 (↑ 0.4) | 91.2 (↑ 0.2) | 91.3 (↑ 0.4) |
| | SGG-PaRot | xyz | 90.8 (↑ 0.2) | 90.7 (↑ 0.1) | 91.1 (↑ 0.3) |

The bold entities indicates best values.

From Table IV, it can be seen that the performance of specialized networks is obviously better than the regular ones, particularly in the $z/SO3$ case. After integrating SGGConvs, the performance of all three baseline networks is improved. On one hand, SGGConvs show significant improvements over the ‘Regular’ baseline networks (including PointNet++ [23] and DGCNN [10]). In all three training/testing settings, the classification accuracy has been improved by at least 0.6%. Particularly in the $z/SO3$ case, the improvements on the classification accuracy all exceed 5%. As a result, the accuracy of SGG-PointNet++ reaches 91.4% in the z/z case, which is comparable to the best-performing SFCNN [57] in the ‘Specialized’ type. In the $SO3/SO3$ case, the accuracy of SGG-DGCNN reaches 90.8%, which is second only to the best-performing RICnv++ [22] and PaRot [40] in the ‘Specialized’ type. On the other hand, SGGConvs also have certain improvements on the ‘Specialized’ type of RICnv++ [22] and PaRot [40], with classification accuracy improved by 0.4%, 0.2%, and 0.4% on RICnv++ [22], and by 0.2%, 0.1%, and 0.3% on PaRot [40] across the three settings. Finally, SGG-RICnv++ achieves the highest classification accuracy in both the $z/SO3$ and $SO3/SO3$ cases. All of the above results demonstrate that SGGConvs can enhance the rotation-invariance of baseline networks, even when

TABLE V
MODEL COMPLEXITY AND THE PERFORMANCE OF DIFFERENT METHODS

| | Method | Params (M) | W/OR | $SO3/SO3$ |
|-------------|-------------------|---------------|---------------------|---------------------|
| Specialized | SphericalCNN [56] | 1.74 | - | 86.9 |
| | RICnv [21] | 0.70 | - | 86.4 |
| | ClusterNet [58] | 1.40 | - | 87.1 |
| | SGMNet [27] | 1.81 | - | 90.0 |
| | RICnv++ [22] | 0.42 | 90.6 | 90.9 |
| | PaRot [40] | 1.63 | 90.5 | 90.8 |
| Regular | PointNet [2] | 3.50 | 89.2 | 80.3 |
| | PointNet++ [23] | 1.74 | 90.7 | 88.5 |
| | PointCNN [9] | 0.60 | 92.2 | 84.5 |
| | DGCNN [10] | 1.81 | 92.9 | 89.2 |
| Ours | SGG-PointNet++ | 1.87 (↑ 0.13) | 92.6 (↑ 1.9) | 89.7 (↑ 1.2) |
| | SGG-DGCNN | 2.06 (↑ 0.25) | 93.7 (↑ 0.8) | 90.8 (↑ 1.6) |
| | SGG-RICnv++ | 0.51 (↑ 0.09) | 91.0 (↑ 0.4) | 91.3 (↑ 0.4) |
| | SGG-PaRot | 1.81 (↑ 0.18) | 90.7 (↑ 0.2) | 91.1 (↑ 0.3) |

‘W/O R’ represents regular training/testing without any rotation transformation on the input data.

the network itself is specifically-designed for improving the rotation-invariance.

B. Model Complexity

1) *Parameters*: In order to measure the extra computational overhead incurred by SGG-Net, we first conduct an evaluation on the number of parameters for each network, as shown in Table V. We use Params (The number of trainable parameters) and OA (Overall Accuracy on ModelNet40 [41]) as metrics to measure the complexity and performance of each model respectively. Note that since some of the methods in Table IV did not provide the number of parameters in the original papers, we do not list them in Table V for comparison.

After integrating SGGConvs, the increased parameters to the four baseline networks are 0.13 M, 0.25 M, 0.09 M, and 0.18 M, with an increase ratio of 7.4%, 13.8%, 21.4%, and 11.0%, respectively. It can be seen that the increase in parameters is small, but the overall classification accuracy has improved significantly. Specifically, SGG-DGCNN and SGG-RICnv++ achieve the highest classification accuracy in the regular training/testing setting (i.e., W/OR) and the $SO3/SO3$ setting, respectively.

2) *FLOPs & Running Time*: To further compare model complexity, we calculate the FLOPs and inference time for the four baseline networks before and after integration with SGG-Net. As shown in Table VI, the computational cost and inference time of each method increase to varying degrees after integration with SGG-Net, primarily due to the SGGConv operations in S-SSGConvMdl and M-SSGConvMdl. The classical PointNet++ [23] and DGCNN [10] have much higher FLOPs compared to the other two newer methods, and accordingly, their SGG-enhanced networks (i.e., SGG-PointNet++ and SGG-DGCNN) also experience a greater increase in FLOPs. However, this does not necessarily mean that the running time will increase proportionally. For example, we can see that the increase in running time for

TABLE VI
FLOPS AND INFERENCE TIME COMPARISON ON MODELNET40. TIME IS MEASURED PER BATCH WITH A SIZE OF 16

| | Method | FLOPs (G) | Inference Time (s) |
|----------|-----------------|-------------------------|-----------------------------|
| Baseline | PointNet++ [23] | 4.07 | 0.1312 |
| | DGCNN [10] | 2.58 | 0.0295 |
| | RICov++ [22] | 0.73 | 0.0474 |
| | PaRot [40] | 2.11 | 0.0332 |
| Ours | SGG-PointNet++ | 4.97 (\uparrow 0.90) | 0.1547 (\uparrow 0.0235) |
| | SGG-DGCNN | 4.15 (\uparrow 1.57) | 0.0444 (\uparrow 0.0149) |
| | SGG-RIConv++ | 0.95 (\uparrow 0.22) | 0.0544 (\uparrow 0.0070) |
| | SGG-PaRot | 2.24 (\uparrow 0.13) | 0.0487 (\uparrow 0.0155) |

TABLE VII
MODEL PERFORMANCE WITH DIFFERENT K IN THE $SO3/SO3$ CASE

| K | SGG-PointNet++ | SGG-DGCNN | SGG-RIConv++ |
|-----|----------------|-------------|--------------|
| 20 | 89.4 | 90.2 | 89.9 |
| 30 | 89.7 | 90.8 | 91.3 |
| 40 | 89.2 | 90.5 | 91.0 |

The bold entities indicates best values.

SGG-DGCNN is actually quite similar to that of SGG-PaRot. Ultimately, the increase in running time across all four baselines is relatively small, with SGG-RIConv++ only increasing by 0.007 s. The above analysis fully demonstrates the efficiency of SGG-Net as a plugin.

C. Ablation Studies on ModelNet40

We conduct further experiments on ModelNet40 [41] to evaluate the performance of our method from three different aspects.

Number of Nearest Neighbors (K): As described in Section III-B, we need to use K nearest neighboring points to construct the Spherical Geometric Descriptor (SGD) for each point, and thus the choice of K can directly affect the performance of SGG-Nets. To quantitatively analyze the influence of K on the network, we conduct experiments on SGG-PointNet++, SGG-DGCNN, and SGG-RIConv++ with different values of K . Table VII shows the overall accuracy (OA) of the classification task on ModelNet40 [41]. The three models all perform best at $K = 30$, and we speculate that this is because a larger K can give the network a larger geometric receptive field, which can capture more global features, but an overly large K may easily lose local information. To maintain the best performance of the network, we set 30 as the default value of K in our experiments.

Number of Points: During training, the default number of input points for each network is 1024. To evaluate the network's robustness to sparse point clouds, we randomly discard some input points at the testing time. PointNet++[23] requires a minimum of 128 points as input, while DGCNN [10] can have a smaller number of points, so we input a minimum of 64 and 128 points to DGCNN [10] and PointNet++[23], respectively.

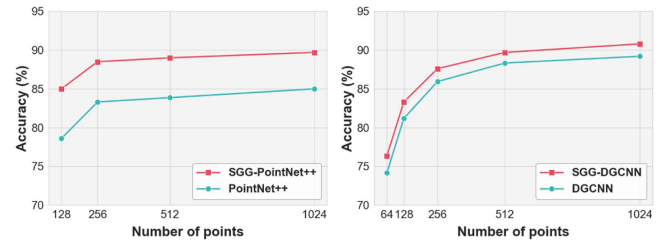


Fig. 7. The classification accuracy on ModelNet40 in the $SO3/SO3$ case after randomly dropping out some input points. Left: PointNet++ and SGG-PointNet++; Right: DGCNN and SGG-DGCNN.

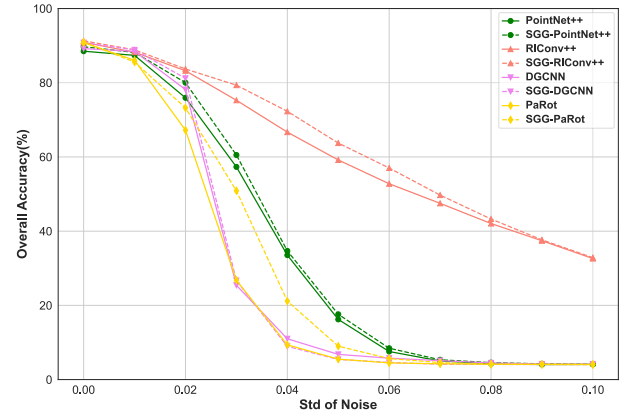


Fig. 8. Robustness test of our SGG-Net against noise. The horizontal and vertical axes represent the standard deviation of the noise and the overall classification accuracy, respectively.

As shown in Fig. 7, as the number of points decreases, the performance of each network declines, but SGG-PointNet++ and SGG-DGCNN always perform better than their baseline versions. Even in the worst case, the classification accuracy of the two networks can still reach 85.3% and 76.8%, respectively.

Robustness to Noise: To test the robustness of our SGG-Net against noise, we inject varying levels of Gaussian noise into the test data and observe the changes in classification accuracy, based on the pre-trained baseline models and their SGG-enhanced counterparts in the $SO3/SO3$ setting. As shown in Fig. 8, as the noise level increases, the classification accuracy of all models gradually decreases. However, we can see that the SGG-enhanced models generally exhibit a smaller decline in accuracy compared to their original baseline models. This is particularly evident with SGG-RIConv++ and SGG-PaRot, which sustain much higher classification accuracy compared to RICov++ [22] and PaRot [40], respectively, especially when the noise standard deviation falls within the range (0.02, 0.06). This clearly demonstrates the robustness of our SGG-Net to data noise.

Importance of SGD Components: In order to evaluate the importance of each component of SGD, we conduct ablation study by removing one component at each time, and the result is shown in Table VIII. We can see that removing any component will cause a decrease in classification performance. Specifically, for SGG-PointNet++, the removal of φ has the biggest impact, while for SGG-DGCNN and SGG-RIConv++, the removal of θ has

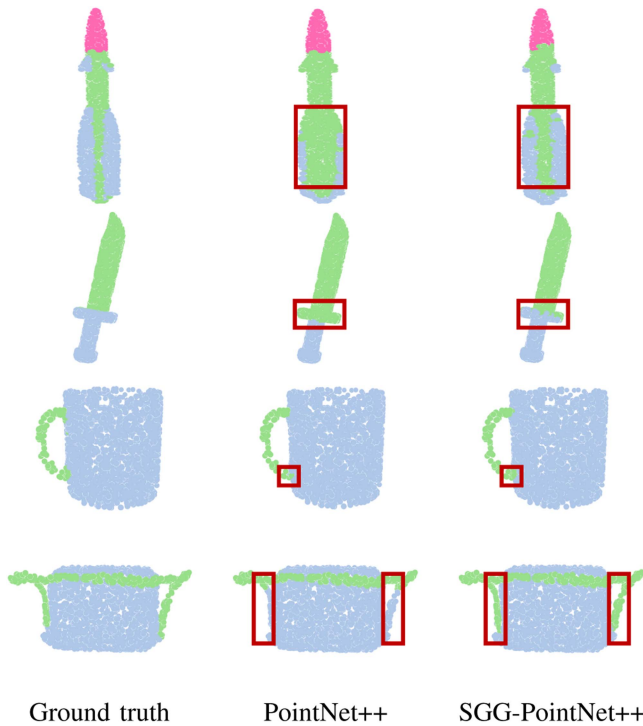


Fig. 9. Visualization of the part segmentation results of PointNet++ and SGG-PointNet++ in the SO_3/SO_3 case. From top to bottom: rocket, knife, mug, and bag; From left to right: Ground truth, PointNet++, and SGG-PointNet++.

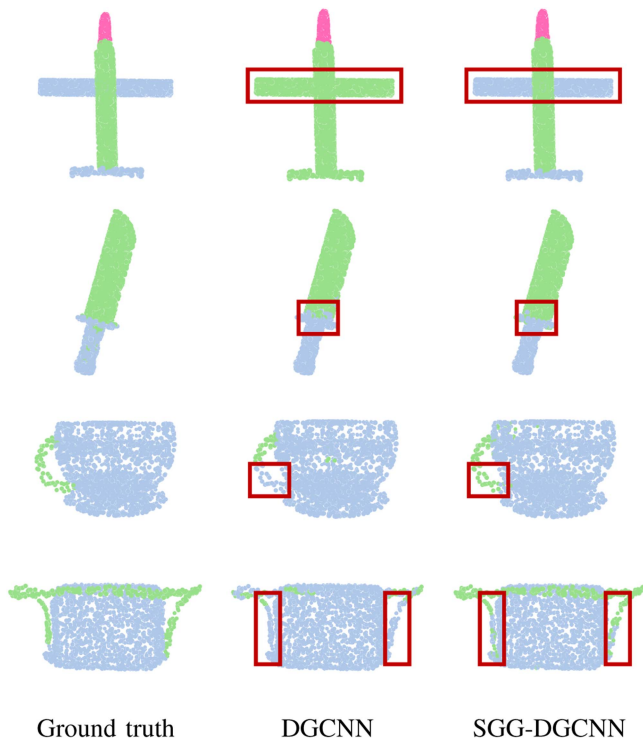


Fig. 10. Visualization of the part segmentation results of DGCNN and SGG-DGCNN in the SO_3/SO_3 case. From top to bottom: aeroplane, knife, mug, and bag; From left to right: Ground truth, DGCNN, and SGG-DGCNN.

TABLE VIII
ABLATION STUDY ON THE COMPONENTS OF SGD IN THE SO_3/SO_3 CASE

| d | θ | φ | SGG-PointNet++ | SGG-DGCNN | SGG-RICConv++ |
|-----|----------|-----------|----------------|-------------|---------------|
| X | X | | 89.7 | 90.8 | 91.3 |
| | | | 89.5 | 89.9 | 91.1 |
| | | | 89.6 | 89.0 | 90.7 |
| | | X | 89.3 | 90.3 | 91.0 |

The bold entities indicates best values.

TABLE IX
CLASSIFICATION RESULTS ON SCANOBJECTNN WITH z/SO_3 ROTATION

| | Method | OBJ_BG | PB_T50_RS |
|-------------|------------------------------|---------------------|---------------------|
| Specialized | RICConv [21] | 78.4 | 68.3 |
| | GCACConv [65] | 78.2 | 70.0 |
| | RICConv++ [†] [22] | 80.6 | 75.1 |
| | SRINet++ [20] | 70.0 | 61.2 |
| | PaRot [40] | 82.1 | 80.9 [†] |
| Regular | PointNet [2] | 16.7 | 17.1 |
| | PointCNN [9] | 14.6 | 14.9 |
| | PointNet++ [†] [23] | 55.5 | 48.6 |
| | DGCNN [†] [10] | 61.1 | 48.9 |
| Ours | SGG-PointNet++ | 61.9 (↑ 6.4) | 49.4 (↑ 0.8) |
| | SGG-DGCNN | 63.8 (↑ 2.7) | 50.2 (↑ 1.3) |
| | SGG-RICConv++ | 83.2 (↑ 2.6) | 76.5 (↑ 1.4) |
| | SGG-PaRot | 85.2 (↑ 3.1) | 81.8 (↑ 0.9) |

the biggest impact. These results fully demonstrate that all three components of SGD have a certain degree of importance to our SGG-Nets.

D. Classification on ScanObjectNN

Data: We further conduct classification task on ScanObjectNN [42], a real-world point cloud dataset captured by RGB-D camera, which includes challenging scenarios such as object occlusion and varying point densities. This dataset presents greater challenges compared to ModelNet40 [41]. The dataset classifies 2,902 models into 15 categories, where we take 2,319 models for training and 583 for testing. It is worth mentioning that the dataset includes five difficulty levels, and in our experiments we select the hardest PB_T50_RS as well as the simplest OBJ_BG with z/SO_3 transformation to evaluate our SGG-Nets.

Implementation: The network architecture is the same as that on ModelNet40 [41], except that the output dimension of the classifier is changed from 40 to 15. Since the original papers of PointNet++ [23] and DGCNN [10] did not provide classification results on ScanObjectNN [42], we test the classification performance of both models using the same experimental configuration as on ModelNet40 [41]. Specifically, all models are trained with a batch size of 32 for 300 epochs.

TABLE X
PART SEGMENTATION RESULTS ON SHAPE NET WITH $z/SO3$ ROTATION

| Method | MEAN | AERO | BAG | CAP | CAR | CHAIR | EAR PHONE | GUITAR | KNIFE | LAMP | LAPTOP | MOTOR | MUG | PISTOL | ROCKET | SKATE BOARD | TABLE |
|------------------------------|---------------------|-------------|-------------|-------------|-------------|-------------|--------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|----------------|-------------|
| SpiderCNN [67] | 42.9 | 48.8 | 47.9 | 41.0 | 25.1 | 59.8 | 23.0 | 28.5 | 49.5 | 45.0 | 83.6 | 20.9 | 55.1 | 41.7 | 36.5 | 39.2 | 41.2 |
| RIConv [21] | 75.3 | 80.6 | 80.0 | 70.8 | 68.8 | 86.8 | 70.3 | 87.3 | 84.7 | 77.8 | 80.6 | 57.4 | 91.2 | 71.5 | 52.3 | 66.5 | 78.4 |
| PRIN [67] | 64.6 | 62.6 | 61.2 | 71.1 | 57.0 | 76.7 | 61.8 | 72.3 | 74.7 | 64.7 | 70.6 | 40.1 | 77.0 | 65.0 | 47.8 | 62.3 | 69.4 |
| Li [60] | 79.2 | 81.4 | 82.3 | 86.3 | 75.3 | 88.5 | 72.8 | 90.3 | 82.1 | 81.3 | 81.9 | 67.5 | 92.6 | 75.5 | 54.8 | 75.1 | 78.9 |
| RIConv++ [†] [22] | 76.4 | 79.8 | 78.5 | 86.9 | 68.2 | 88.1 | 61.3 | 88.6 | 78.9 | 80.3 | 92.5 | 52.7 | 91.1 | 67.8 | 59.1 | 69.9 | 78.9 |
| PaRot [40] | 79.2 | 82.7 | 79.2 | 82.3 | 75.3 | 89.4 | 73.9 | 91.1 | 85.6 | 81.0 | 79.5 | 65.3 | 93.9 | 79.2 | 55.0 | 72.4 | 79.5 |
| PointNet [2] | 37.8 | 40.4 | 48.1 | 46.3 | 24.5 | 45.1 | 39.4 | 29.2 | 42.6 | 52.7 | 36.7 | 21.2 | 55.0 | 29.7 | 26.6 | 32.1 | 35.8 |
| PointNet++ [†] [23] | 45.4 | 27.8 | 60.1 | 67.4 | 28.3 | 46.8 | 31.1 | 67.3 | 60.9 | 66.2 | 37.7 | 23.4 | 51.3 | 27.5 | 32.2 | 39.2 | 59.2 |
| PointCNN [9] | 34.7 | 21.8 | 52.0 | 52.1 | 23.6 | 29.4 | 18.2 | 40.7 | 36.9 | 51.1 | 33.1 | 18.9 | 48.0 | 23.0 | 27.7 | 38.6 | 39.9 |
| DGCNN [†] [10] | 49.7 | 29.8 | 50.2 | 75.2 | 34.0 | 43.9 | 35.4 | 60.6 | 74.7 | 62.1 | 45.1 | 25.7 | 55.4 | 47.7 | 40.4 | 49.0 | 66.3 |
| SGG-PointNet++ | 49.2 (↑ 3.8) | 31.3 | 69.2 | 66.1 | 30.9 | 56.7 | 35.0 | 70.0 | 69.1 | 70.9 | 42.9 | 22.3 | 49.1 | 35.3 | 31.7 | 47.1 | 60.4 |
| SGG-DGCNN | 59.9 (↑ 10.2) | 31.0 | 68.5 | 80.8 | 36.6 | 61.2 | 61.9 | 86.0 | 83.3 | 78.1 | 48.4 | 38.0 | 61.3 | 61.4 | 44.1 | 50.5 | 67.3 |
| SGG-RIConv++ | 79.5 (↑ 3.1) | 81.6 | 80.4 | 87.3 | 74.2 | 89.6 | 65.5 | 91.2 | 82.2 | 81.1 | 93.7 | 63.8 | 92.9 | 74.5 | 59.6 | 73.6 | 80.9 |
| SGG-PaRot | 79.9 (↑ 0.7) | 83.4 | 79.6 | 83.2 | 75.5 | 89.7 | 76.2 | 91.3 | 85.7 | 81.3 | 82.8 | 66.4 | 94.0 | 79.6 | 55.6 | 73.7 | 79.7 |

The bold entities indicates best values.

TABLE XI
PART SEGMENTATION RESULTS ON SHAPE NET WITH $SO3/SO3$ ROTATION

| Method | MEAN | AERO | BAG | CAP | CAR | CHAIR | EAR PHONE | GUITAR | KNIFE | LAMP | LAPTOP | MOTOR | MUG | PISTOL | ROCKET | SKATE BOARD | TABLE |
|------------------------------|---------------------|-------------|-------------|-------------|-------------|-------------|--------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|----------------|-------------|
| SpiderCNN [66] | 72.3 | 74.3 | 72.4 | 72.6 | 58.4 | 82.0 | 68.5 | 87.8 | 81.3 | 71.3 | 94.5 | 45.7 | 88.1 | 83.4 | 50.5 | 60.8 | 78.3 |
| RIConv [21] | 75.5 | 80.6 | 80.2 | 70.7 | 68.8 | 86.8 | 70.4 | 87.2 | 84.3 | 78.0 | 80.1 | 57.3 | 91.2 | 71.3 | 52.1 | 66.6 | 78.5 |
| PRIN [67] | 67.6 | 67.4 | 61.5 | 69.7 | 59.5 | 77.7 | 65.8 | 75.7 | 77.2 | 65.9 | 82.0 | 44.2 | 79.8 | 63.6 | 53.0 | 67.5 | 70.7 |
| Li [60] | 79.4 | 81.4 | 84.5 | 85.1 | 75.0 | 88.2 | 72.4 | 90.7 | 84.4 | 80.3 | 84.0 | 68.8 | 92.6 | 76.1 | 52.1 | 74.1 | 80.0 |
| RIConv++ [†] [22] | 79.0 | 81.2 | 75.4 | 86.3 | 73.9 | 89.3 | 71.1 | 90.4 | 82.4 | 81.1 | 94.0 | 55.4 | 92.7 | 76.4 | 61.0 | 71.3 | 80.7 |
| PaRot [40] | 79.5 | 82.9 | 82.1 | 83.2 | 75.7 | 89.4 | 76.1 | 91.5 | 86.1 | 81.4 | 80.3 | 59.3 | 94.3 | 79.7 | 57.0 | 73.3 | 79.2 |
| PointNet [2] | 74.4 | 81.6 | 68.7 | 74.0 | 70.3 | 87.6 | 68.5 | 88.9 | 80.0 | 74.9 | 83.6 | 56.5 | 77.6 | 75.2 | 53.9 | 69.4 | 79.9 |
| PointNet++ [†] [23] | 76.4 | 78.8 | 71.9 | 87.6 | 70.1 | 88.9 | 65.5 | 88.9 | 77.9 | 78.7 | 93.8 | 52.2 | 92.8 | 76.8 | 51.1 | 67.7 | 80.5 |
| PointCNN [9] | 71.4 | 78.0 | 80.1 | 78.2 | 68.2 | 81.2 | 70.2 | 82.0 | 70.6 | 68.9 | 80.8 | 48.6 | 77.3 | 63.2 | 50.6 | 63.2 | 82.0 |
| DGCNN [†] [10] | 72.5 | 77.5 | 72.5 | 76.3 | 54.1 | 86.5 | 66.4 | 87.8 | 83.1 | 80.3 | 81.6 | 35.8 | 85.4 | 76.5 | 50.2 | 66.1 | 80.1 |
| SGG-PointNet++ | 79.3 (↑ 2.9) | 81.0 | 77.4 | 83.9 | 72.3 | 88.6 | 73.2 | 90.0 | 84.5 | 83.0 | 94.5 | 59.0 | 93.9 | 74.9 | 60.0 | 71.9 | 80.9 |
| SGG-DGCNN | 81.8 (↑ 9.3) | 84.2 | 79.4 | 83.2 | 78.9 | 90.5 | 75.5 | 91.3 | 87.0 | 82.0 | 95.8 | 71.8 | 94.8 | 82.2 | 57.2 | 73.9 | 81.0 |
| SGG-RIConv++ | 79.6 (↑ 0.6) | 82.8 | 78.2 | 83.1 | 77.3 | 89.7 | 68.6 | 90.8 | 82.8 | 81.6 | 95.5 | 59.8 | 91.8 | 76.7 | 60.8 | 73.1 | 81.6 |
| SGG-PaRot | 81.0 (↑ 1.5) | 84.2 | 82.7 | 84.6 | 79.4 | 90.6 | 77.2 | 91.7 | 87.4 | 81.5 | 80.6 | 60.7 | 94.9 | 83.5 | 61.4 | 74.7 | 80.6 |

The bold entities indicates best values.

Results: The classification results on ScanObjectNN [42] are shown in Table IX. As we discussed earlier, the performance of regular networks on rotation-invariance is usually not as good as that of specialized networks. By integrating SGGConvs, the performance of regular networks has been greatly improved. Specifically, SGG-PointNet++ is improved by 6.4% and 0.8% on the two difficulty levels, respectively, compared to its baseline version, while SGG-DGCNN is improved by 2.7% and 1.3%, respectively. In addition, for specialized networks like RIConv++ [22] and PaRot [40] that already have outstanding performance in dealing with rotation-invariance, accuracy is also further improved after equipped with SGG-Convs. Specifically, SGG-RIConv++ has improved the accuracy by 2.6% and 1.4% respectively compared to RIConv++ [22] at the two levels. Similarly, SGG-PaRot has also improved its classification accuracy by 3.1% and 0.9% respectively compared to PaRot [40], ultimately achieving state-of-art performance in both levels. These results indicate that even on complex real-world datasets with challenging scenarios, our proposed SGG-Net can effectively enhance the rotation invariance of the baseline network.

E. Part Segmentation on ShapeNet

Data: We conduct part segmentation on ShapeNet [43], which contains 16,880 CAD models from 16 object categories. Each model is labeled with 2 to 6 parts, and the task is to predict a part label for each point of the input point cloud. Following Zhang et al. [22], we set 14,006 models for training and 2874 models for testing. We sample 2048 points from each training shape, and report the classification accuracy under two train/test settings: $z/SO3$ and $SO3/SO3$.

Implementation: As before, for all the networks the training configuration follows the original papers. However, for fairness we train each model with a batch size of 16 for 200 epochs.

Results: Tables X and XI present the part segmentation results on ShapeNet [43] in the $z/SO3$ and $SO3/SO3$ settings respectively. We chose mIoU (%) as the evaluation metric and calculate the average mIoU of all the categories.

From Table X, we can see that with the integration of SGG-Convs, the average mIoU of SGG-PointNet++ has increased by 3.8% compared to PointNet++ [23], while SGG-DGCNN

has increased by 10.2% compared to DGCNN [10], showing considerable improvements. The average mIoU of the original RConv++ [22] is lower than that of the Li et al. [60] and PaRot [40], but after integrating SGGConvs, SGG-RConv++ surpasses them. Additionally, after integrating with SGG-Net, PaRot [40] achieves an improvement of 0.7% in accuracy, ultimately reaching the highest performance at 79.9%.

From Table XI, it can be seen that under the SO_3/SO_3 rotation, the average mIoU of the four baseline networks are increased by 2.9%, 9.3%, 0.6%, and 1.5% respectively after the integration of our SGGConvs. Interestingly, SGGConvs have a significant improvement over DGCNN [10], and the final average mIoU of SGG-DGCNN even exceeds that of SGG-RConv++ and SGG-PaRot, achieving the best performance.

Besides, we choose PointNet++ [23] and DGCNN [10] to visualize and compare their segmentation results under SO_3/SO_3 rotation before and after equipping with SGG-Net. The results are shown in Figs. 9 and 10. It can be seen that the baseline networks equipped with SGGConvs have more accurate segmentation results on the boundaries of parts, which are closer to ground-truth.

V. CONCLUSIONS AND FUTURE WORK

In this article, we propose a series of generic plugin networks called SGG-Nets that is obtained by integrating the proposed SGGConvs into existing point cloud networks to promote their performance. SGG-Nets can not only enhance the rotation-invariance of the baseline network, but also improve its performance on various point cloud analysis tasks, e.g., classification and segmentation. Furthermore, because of its simplicity, the additional overhead caused by SGGConvs is very small. The conducted experiments have shown the effectiveness and efficiency of SGG-Nets. As for future work, the idea of manually constructing rotation-invariant feature descriptors by establishing a local spherical coordinate system may also be used for other complex 3D data structures besides point cloud data, such as 3D mesh, and we will try this in the future.

REFERENCES

- [1] Y. Guo et al., "Deep learning for 3D point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4338–4364, Dec. 2021.
- [2] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.
- [3] X.-F. Han, Y.-F. Jin, H.-X. Cheng, and G.-Q. Xiao, "Dual transformer for point cloud analysis," *IEEE Trans. Multimedia*, vol. 25, pp. 5638–5648, 2023.
- [4] J. Li, J. Wang, and T. Xu, "PointGL: A simple global-local framework for efficient point cloud analysis," *IEEE Trans. Multimedia*, vol. 26, pp. 6931–6942, 2024.
- [5] G. Zhu, Y. Zhou, R. Yao, and H. Zhu, "Cross-class bias rectification for point cloud few-shot segmentation," *IEEE Trans. Multimedia*, vol. 25, pp. 9175–9188, 2023.
- [6] Y. Zhang, L. Zhang, X. Zhao, H. Fu, and D. Yu, "Automatic point cloud registration for 3D virtual-to-real registration using macro and micro structures," *IEEE Trans. Multimedia*, vol. 26, pp. 6566–6581, 2024.
- [7] L. Tan et al., "Projected generative adversarial network for point cloud completion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 2, pp. 771–781, Feb. 2022.
- [8] T. Huang et al., "Adaptive recurrent forward network for dense point cloud completion," *IEEE Trans. Multimedia*, vol. 25, pp. 5903–5915, 2023.
- [9] Y. Li et al., "PointCNN: Convolution on x-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 828–838.
- [10] Y. Wang et al., "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.
- [11] M.-H. Guo et al., "PCT: Point cloud transformer," *Comput. Vis. Media*, vol. 7, no. 2, pp. 187–199, 2021.
- [12] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 16259–16268.
- [13] X. Deng, W. Zhang, Q. Ding, and X. Zhang, "PointVector: A vector representation in point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 9455–9465.
- [14] J. Fei and Z. Deng, "Rotation invariance and equivariance in 3D deep learning: A survey," *Artif. Intell. Rev.*, vol. 57, no. 7, 2024, Art. no. 168.
- [15] S. Kim, J. Park, and B. Han, "Rotation-invariant local-to-global representation learning for 3D point cloud," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 8174–8185.
- [16] Z. Xiao et al., "Endowing deep 3D models with rotation invariance based on principal component analysis," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2020, pp. 1–6.
- [17] J. Zhang, M.-Y. Yu, R. Vasudevan, and M. Johnson-Roberson, "Learning rotation-invariant representations of point clouds using aligned edge convolutional neural networks," in *Proc. Int. Conf. 3D Vis.*, 2020, pp. 200–209.
- [18] Y. Zhao et al., "Quaternion equivariant capsule networks for 3D point clouds," in *Proc. 16th Eur. Conf. Comput. Vis.*, Glasgow, U.K., 2020, pp. 1–19.
- [19] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 998–1005.
- [20] X. Sun, Z. Lian, and J. Xiao, "SRINet: Learning strictly rotation-invariant representations for point cloud classification and segmentation," in *Proc. 27th ACM Int. Conf. Multimedia*, 2019, pp. 980–988.
- [21] Z. Zhang, B.-S. Hua, D. W. Rosen, and S.-K. Yeung, "Rotation invariant convolutions for 3D point clouds deep learning," in *Proc. Int. Conf. 3D Vis.*, 2019, pp. 204–213.
- [22] Z. Zhang, B.-S. Hua, and S.-K. Yeung, "RConv: Effective rotation invariant convolutions for 3D point clouds deep learning," *Int. J. Comput. Vis.*, vol. 130, no. 5, pp. 1228–1243, 2022.
- [23] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5105–5114.
- [24] C. Park, Y. Jeong, M. Cho, and J. Park, "Fast point transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16949–16958.
- [25] J. Shu et al., "CFSA-Net: Efficient large-scale point cloud semantic segmentation based on cross-fusion self-attention," *Comput., Mater. Continua*, vol. 77, no. 3, pp. 2677–2697, 2023.
- [26] G. Zhu, Y. Zhou, R. Yao, H. Zhu, and J. Zhao, "Cyclic self-attention for point cloud recognition," *ACM Trans. Multimedia Comput., Commun. Appl.*, vol. 19, no. 1s, pp. 1–19, 2023.
- [27] J. Xu, X. Tang, Y. Zhu, J. Sun, and S. Pu, "SGMNet: Learning rotation-invariant point cloud representations via sorted gram matrix," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 10468–10477.
- [28] Z. Kuang, J. Yu, S. Zhu, Z. Li, and J. Fan, "Effective 3-D shape retrieval by integrating traditional descriptors and pointwise convolution," *IEEE Trans. Multimedia*, vol. 21, no. 12, pp. 3164–3177, Dec. 2019.
- [29] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
- [30] P. Veličković et al., "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018.
- [31] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?," in *Proc. Int. Conf. Learn. Representations*, 2022.
- [32] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4548–4557.
- [33] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "PointWeb: Enhancing local neighborhood features for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5565–5573.
- [34] W. Shi and R. Rajkumar, "Point-GNN: Graph neural network for 3D object detection in a point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1711–1719.
- [35] Z.-H. Lin, S.-Y. Huang, and Y.-C. F. Wang, "Convolution in the cloud: Learning deformable kernels in 3D graph convolution networks for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1800–1809.

- [36] X. Wei, R. Yu, and J. Sun, "View-GCN: View-based graph convolutional network for 3D shape analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1850–1859.
- [37] Y. Liu et al., "Spherical message passing for 3D molecular graphs," in *Proc. Int. Conf. Learn. Representations*, 2022.
- [38] Q. He, Z. Wang, H. Zeng, Y. Zeng, and Y. Liu, "SVGA-Net: Sparse voxel-graph attention network for 3D object detection from point clouds," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 870–878.
- [39] W. Du et al., "A new perspective on building efficient and expressive 3D equivariant graph neural networks," *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36.
- [40] D. Zhang, J. Yu, C. Zhang, and W. Cai, "PaRot: Patch-wise rotation-invariant network via feature disentanglement and pose restoration," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 3418–3426.
- [41] Z. Wu et al., "3D shapenets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1912–1920.
- [42] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1588–1597.
- [43] L. Yi et al., "A scalable active framework for region annotation in 3D shape collections," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 1–12, 2016.
- [44] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Fine-grained 3D shape classification with hierarchical part-view attentions," *IEEE Trans. Image Process.*, vol. 30, pp. 1744–1758, 2021.
- [45] I. Armeni et al., "3D semantic parsing of large-scale indoor spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 1534–1543.
- [46] J. Li, B. M. Chen, and G. H. Lee, "SO-net: Self-organizing network for point cloud analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9397–9406.
- [47] J. Yang et al., "Modeling point clouds with self-attention and gumbel subset sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3323–3332.
- [48] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9621–9630.
- [49] Y. Liu et al., "DensePoint: Learning densely contextual representation for efficient point cloud processing," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 5239–5248.
- [50] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8895–8904.
- [51] H. Thomas et al., "KPConv: Flexible and deformable convolution for point clouds," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 6411–6420.
- [52] W. Han, C. Wen, C. Wang, X. Li, and Q. Li, "Point2Node: Correlation learning of dynamic-node for point cloud feature modeling," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 10925–10932.
- [53] S. Cheng, X. Chen, X. He, Z. Liu, and X. Bai, "PRA-Net: Point relation-aware network for 3D point cloud analysis," *IEEE Trans. Image Process.*, vol. 30, pp. 4436–4448, 2021.
- [54] Z. Yang, L. Jiang, Y. Sun, B. Schiele, and J. Jia, "A unified query-based paradigm for point cloud understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8541–8551.
- [55] Y. Liu, B. Tian, Y. Lv, L. Li, and F. Wang, "Point cloud classification using content-based transformer via clustering in feature space," *IEEE/CAA J. Automatica Sinica*, vol. 10, no. 8, pp. 1–9, 2023.
- [56] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, "Learning SO(3) equivariant representations with spherical CNNs," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 52–68.
- [57] Y. Rao, J. Lu, and J. Zhou, "Spherical fractal convolutional neural networks for point cloud recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 452–460.
- [58] C. Chen et al., "ClusterNet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4994–5002.
- [59] R. Gu, Q. Wu, H. Xu, W. W. Ng, and Z. Wang, "Learning efficient rotation representation for point cloud via local-global aggregation," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2021, pp. 1–6.
- [60] X. Li et al., "A rotation-invariant framework for deep point cloud analysis," *IEEE Trans. Vis. Comput. Graph.*, vol. 28, no. 12, pp. 4503–4514, Dec. 2022.
- [61] J. Yu, C. Zhang, and W. Cai, "Rethinking rotation invariance with point cloud registration," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 3313–3321.
- [62] D. Maturana and S. Scherer, "Voxnet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 922–928.
- [63] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 945–953.
- [64] C. R. Qi et al., "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 5648–5656.
- [65] Z. Zhang, B.-S. Hua, W. Chen, Y. Tian, and S.-K. Yeung, "Global context aware convolutions for 3D point cloud understanding," in *Proc. Int. Conf. 3D Vis.*, 2020, pp. 210–219.
- [66] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 87–102.
- [67] Y. You et al., "Pointwise rotation-invariant network with adaptive sampling and 3D spherical voxel convolution," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 12717–12724.



Jian Zhu received the Bachelor's degree in mathematics from Sun Yat-sen University, Guangzhou, China, in 2005, and the Ph.D. degree in computer science in 2013 from University of Macau, Macao, China. He is currently an Associate Professor and the Associate Dean with the School of Computer Science and Technology, the Guangdong University of Technology, Guangzhou, China. His research interests include machine learning, computer vision, computer graphics.



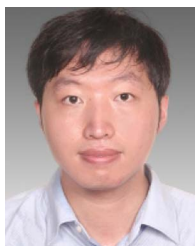
Jianrong Yan received the B.Sc. (Hons.) degree in computer science from the Guangdong University of Technology, Guangzhou, China in 2024. He is currently working toward the master's degree with the School of Computer Science, Fudan University Shanghai, China. His research interests include point cloud processing and analysis related problems, e.g., point cloud classification, registration, and segmentation and detection, in the field of 3D computer vision and deep learning.



Jiebin Huang received the B.Sc. (Hons.) degree in computer science from the Guangdong University of Technology, Guangzhou, China in 2024. He is currently working toward the master's degree in the School of Computer Science and Engineering, South China University of Technology Guangzhou, China. His research interests include computer vision and deep learning.



Yongwei Nie (Member, IEEE) received the B.Sc. and Ph.D. degrees from Computer School, Wuhan University, Wuhan, China in 2009 and 2015, respectively. He is currently an Associate Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His research interests include image and video editing, and computational vision.



Bin Sheng received the B.A. degree in english and the B.Eng. degree in computer science, from the Huazhong University of Science and Technology, China, and the M.Sc. degree in software engineering from the University of Macau, Macau, and the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong. He is currently a Full Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include image/video processing, virtual reality, machine learning, and computer graphics. He is an Associate Editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.



Tong-Yee Lee (Senior Member, IEEE) received the Ph.D. degree in computer engineering from Washington State University, Pullman, WSU, USA in May 1995. He is currently a Chair Professor with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan. He leads the Computer Graphics Group, Visual System Laboratory, National Cheng Kung University (<http://graphics.csie.ncku.edu.tw>). His current research interests include computer graphics, non-photorealistic rendering, medical visualization, virtual reality, and media resizing. He is a Member of the ACM, and an Associate Editor for the IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS (TVCG).