11-2020

Global context aware convolutions for 3D point cloud understanding

Zhiyuan ZHANG
Singapore Management University, zhiyuanzhang@smu.edu.sg

Binh-Son HUA

Wei CHEN

Yibin TIAN

Sai-Kit YEUNG

Global Context Aware Convolutions for 3D Point Cloud Understanding

Zhiyuan Zhang[1] Binh-Son Hua[2,3,*] Wei Chen[1] Yibin Tian[1,*] Sai-Kit Yeung[4,*]

[1]Litemaze Technology [2]VinAI Research, Vietnam [3]VinUniversity, Vietnam
[4]Hong Kong University of Science and Technology

{cszyzhang, binhson.hua}@gmail.com, {chen.wei, tian.yibin}@litemaze.com, saikit@ust.hk

Abstract

Recent advances in deep learning for 3D point clouds have shown great promises in scene understanding tasks thanks to the introduction of convolution operators to consume 3D point clouds directly in a neural network. Point cloud data, however, could have arbitrary rotations, especially those acquired from 3D scanning. Recent works show that it is possible to design point cloud convolutions with rotation invariance property, but such methods generally do not perform as well as translation-invariant only convolution. We found that a key reason is that compared to point coordinates, rotation-invariant features consumed by point cloud convolution are not as distinctive. To address this problem, we propose a novel convolution operator that enhances feature distinction by integrating global context information from the input point cloud to the convolution. To this end, a globally weighted local reference frame is constructed in each point neighborhood in which the local point set is decomposed into bins. Anchor points are generated in each bin to represent global shape features. A convolution can then be performed to transform the points and anchor features into final rotation-invariant features. We conduct several experiments on point cloud classification, part segmentation, shape retrieval, and normals estimation to evaluate our convolution, which achieves state-of-the-art accuracy under challenging rotations.

1. Introduction

Scene understanding has long been a challenging problem in computer vision. Recently, there have been significant advances in applying deep learning [16] to train neural networks for numerous tasks such as object classification and semantic segmentation. With the wide availability of consumer-grade depth sensors, acquiring 3D data has become more intuitive and robust with many 3D datasets available [35, 4, 12, 6, 2, 38, 31]. This leads to increased interests in tackling scene understanding in the 3D domain.

Among the representations for 3D data, a promising direction is to let neural networks consume point cloud data directly since point cloud data is the common data format acquired from depth sensors such as RGB-D or LiDAR cameras. However, since a point cloud is a mathematical set and so it fundamentally differs from an image, passing a point cloud to a traditional neural network like those in the image domain does not work. In principle, it is necessary to design a convolution-equivalent operator in the 3D domain that can take a point cloud as input and output its per-point features. Several attempts have been made with promising results [22, 24, 13, 17, 37, 41].

Despite such research efforts, a problem often overlooked in point cloud convolution is that the operator does not exhibit rotation invariance. A viable solution in 2D deep learning is to augment training data with random rotations. However, in 3D, such data augmentation becomes less effective due to the additional degree of freedom in representing 3D rotations, which can make training prohibitively expensive. A few works turn to learn rotation-invariant features [40, 25, 21, 7, 5], which allows consistent predictions given arbitrarily rotated point clouds.

Unfortunately, a limitation from previous works is that rotation-invariant convolution does not yield features that are as distinctive as translation-invariant convolution. This makes performing object classification with aligned data more accurate than performing the same task with data with arbitrary rotations. For exact rotation invariance, it is expected that the rotation-invariant convolution is as accurate as its translation-invariant sibling.

In this paper, we propose a novel approach to perform rotation-invariant convolution for point clouds. Our key observation is that when rotation invariance is added, it introduces some ambiguities and thus reduces feature distinctiveness. To address this problem, we propose to integrate global context information from the input point cloud to the convolution, resulting in a global context aware convolution

*Corresponding author

2475-7888/20/$31.00 ©2020 IEEE
DOI 10.1109/3DV50981.2020.00031

for 3D point clouds. The main contributions of this work are:

- GCAConv, a novel rotation-invariant convolution operator that output features from local point sets and global anchors. Each anchor is built from subdivided spaces using a globally-weighted local reference frame at each keypoint. By explicit encoding the relation between local point sets and the global anchors, GCAConv can capture both local and global context;
- GCANet, a neural network architecture that uses GCAConv for learning rotation-invariant features for 3D point clouds. The network allows consistent performance across training/testing scenarios that involves different rotation modes;
- Applications of GCANet on object classification, object part segmentation, shape retrieval, and normals estimation that achieve the state-of-the-art performance under challenging rotations.

2. Related Works

Deep learning in the 2D domain has witnessed great success in solving scene understanding tasks such as object classification, semantic segmentation, normal estimation, etc. Drawing from this inspiration, techniques for deep learning in the 3D domain has recently been developed with promising results. In this section, we review the state-of-the-art research in deep learning with 3D data, and then focus on techniques that enable feature learning on point clouds for scene understanding tasks.

Early research in 3D deep learning focus on regular and structured representations of 3D scenes such as multiple 2D images [28, 23, 9], 3D volumes [23, 18], hierarchical data structures like octree [26] or kd-trees [15, 33]. Such representations yield good performance. However, they face challenges from a practical point of view due to memory consumption, imprecise representation, or lack of scalability when high-resolution data is employed.

Many recent works in 3D deep learning switched to investigate how to learn with 3D point cloud, a more compact and intuitive representation compared to volumes and image sets. However, performing deep learning with 3D point clouds is not as straightforward as extending 2D image convolution to 3D because mathematically, a point cloud is a set. To define a valid convolution for a point cloud, it is necessary to ensure that the output features from a convolution is invariant to the permutation of the point set. PointNet [22] pioneered such a solution to output global features by max-pooling per-point features from MLPs. Several follow-up works focus on designing convolutions that can learn local features for a point cloud efficiently [13, 24, 17, 37, 34, 41]. Please also refer to the technical report by Guo et al. [11] for further summary of many deep learning techniques for 3D point clouds.

A fundamental missing feature in the previously mentioned convolution for point clouds is that rotation invariance is not supported. A common solution is to augment the training data with arbitrary rotations, but a limitation of doing so is that generalizing the predictions to unseen rotations is challenging, not mentioning that the training time becomes longer due to the increased amount of training data. Instead, it is desirable to have a point cloud convolution with rotation-invariant features.

To this end, Rao et al. [25] map a point cloud to a spherical domain to define a rotation-invariant convolution. Zhang et al. [40] proposed a convolution that operates on features built from Euclidean distances and angles. Poulenard et al. [21] proposed to integrate spherical harmonics to a convolution. You et al. [39] transform the point cloud onto spherical voxel grids and apply convolution in the transformed domain. A great benefit of such techniques is that it allows *consistent* predictions across training/testing scenarios with or without rotations being applied to the data, and they can generalize robustly to inputs with unseen rotations. Despite that, so far these techniques share a common limitation: their performance is inferior to that in translation-invariant point cloud convolution. A typical example is the accuracy in object classification task on ModelNet40 dataset [35]. State-of-the-art techniques such as PointNet [22], PointNet++ [24], PointCNN [17], or ShellNet [41] report between 89% to 93% of accuracy while techniques with rotation-invariant convolution only report up to 86% of accuracy [40, 21]. Our work in this paper is dedicated to analyze and address this problem.

3. Background

Let us first analyze the performance of existing point cloud convolutions and their rotation-invariant counterparts. We select object classification task as the key task for our analysis. An observation is that the classification accuracy drops when rotation-invariant convolution is applied. We further dissect this phenomenon by visualizing the latent space learnt by the neural networks using t-SNE [32]. The results are shown in Figure 1.

In this figure, we follow Esteves et al. [8] and Zhang et al. [40] to evaluate three scenarios for object classification: z/z, SO3/SO3, and z/SO3. In case z/z, we use data augmented with rotation about gravity axis for training and testing. In case SO3/SO3, we use data augmented with arbitrary rotations for training and testing. In case z/SO3, we train with data by z-rotations and test with data by SO3 rotations. The first scenario has been extensively evaluated by previous point cloud convolution methods. The second and third scenario is specially designed to evaluate rotation invariance. The third scenario is the most challenging as it is

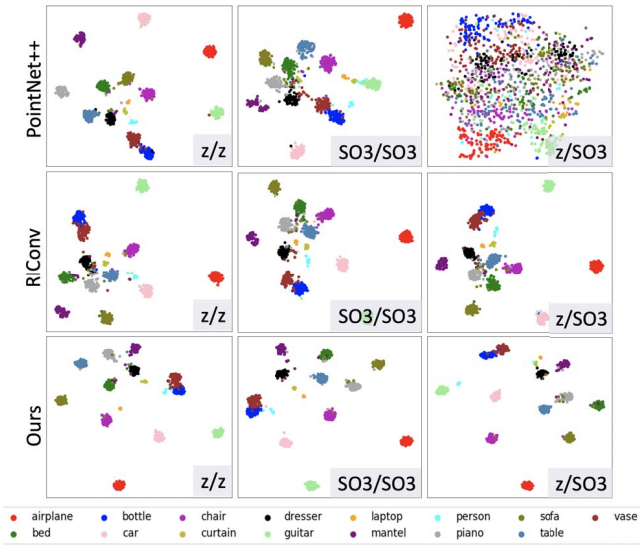

Figure 1. t-SNE comparisons of the latent features for PointNet++ [24], RIConv [40], and our method under three different rotation settings. The clusters in the t-SNEs show that to make good decisions in object classification, it is desirable to have the cluster boundaries as separated as possible.

designed to test whether a convolution can generalize well to unseen rotations.

As can be seen, latent space learnt by rotation-invariant convolution such as RIConv by Zhang et al. [40] does not exhibit good discrimination among classes. The main difference between such convolution and traditional point cloud convolution is that it no longer works with point coordinates at start. In the case of RIConv, the points are transformed into Euclidean based features including distances and angles, which are not as unique as point coordinates since many points can share the same distance and angles. This is well reflected into the t-SNE in the first column (z/z) in Figure 1. PointNet++ [24] has a good separation among the clusters while RIConv [40] has more condensed clusters in the center, resulting in more ambiguities during classification.

Similarly, in the second column (SO3/SO3), PointNet++ and RIConv has similar clustering, which explains their similar performance in the classification (see more quantitative comparisons in Table 1). Finally, the third column (z/SO3) highlights the strength of rotation-invariant convolutions as they can still maintain consistent predictions and generalize well to unseen conditions. In this case, the t-SNEs show that PointNet++ cannot generalize effectively.

The goal of our work is to devise a convolution that can output highly distinctive rotation-invariant features. Here we achieve this by introducing features from a global context to design a new rotation-invariant convolution. We are inspired by the fact that for each point in a point cloud, its 3D coordinates encode global information. Such global information is lost when one converts the coordinates into some rotation-invariant features such as distance and angles as done by Zhang et al. [40].

4. Our Method

Our rotation-invariant convolution is built upon two key concepts: a repeatable and robust local reference frame and a global context using anchors. The idea of using local reference frames is related to spatial transformer [14] which is also leveraged by PointNet [22]. However, as spatial transformer is data-driven, it does not work well to unseen conditions such as the z/SO3 test in Figure 1. To achieve robustness, we build local reference frames (LRFs) at the keypoints of the point cloud so that features can be learnt in such local spaces. At a keypoint, not only points in its local neighborhood can strongly affect the construction of the reference frame, but non-neighboring points can also contribute to such construction. It is well known that repeatable and robust LRFs are keys to traditional 3D point descriptors [30].

After the LRFs are constructed, theoretically we can simply proceed to learn features of the local point sets. However, as previously mentioned, global shape information are also useful for feature learning. We also retain such global information and integrate them into the convolution. Here we achieve this through *anchors*. Each anchor is defined as a representative point in each subspace formed by the axes of the LRF. Given a LRF, it is possible to construct eight subspaces. At each LRF, the anchors thus approximate global features of the point cloud and we integrate such features to define our convolution.

4.1. Globally Weighted Local Reference Frames

For an input point set, we use farthest point sampling to select a set of keypoints which can fully cover the underlying point cloud and denoted as Q. For each keypoint $p \in Q$, we use it as a query to obtain local region Ω_p centroid at p. We wish to use deep learning to extract rotation invariant features from the local region. To begin with local features learning, it is necessary to construct a local reference frame (LRF) such that the 3D coordinates can be transformed into rotation invariant features. The unit vectors of the LRF at p can be determined by normalizing the eigenvectors of the covariance matrix

$$\Sigma_p = \sum_{i=1}^{N_{sub}} (x_i - p)(x_i - p)^\top, \tag{1}$$

where N_{sub} is the number of points in the local region and $x_i \in \Omega_p$. However, the LRF via such computation is unstable and sensitive to noise. Slight point variations can affect the LRF and make it not repeatable. Moreover, when a local region Ω_p undergoes some rotations, ambiguity can

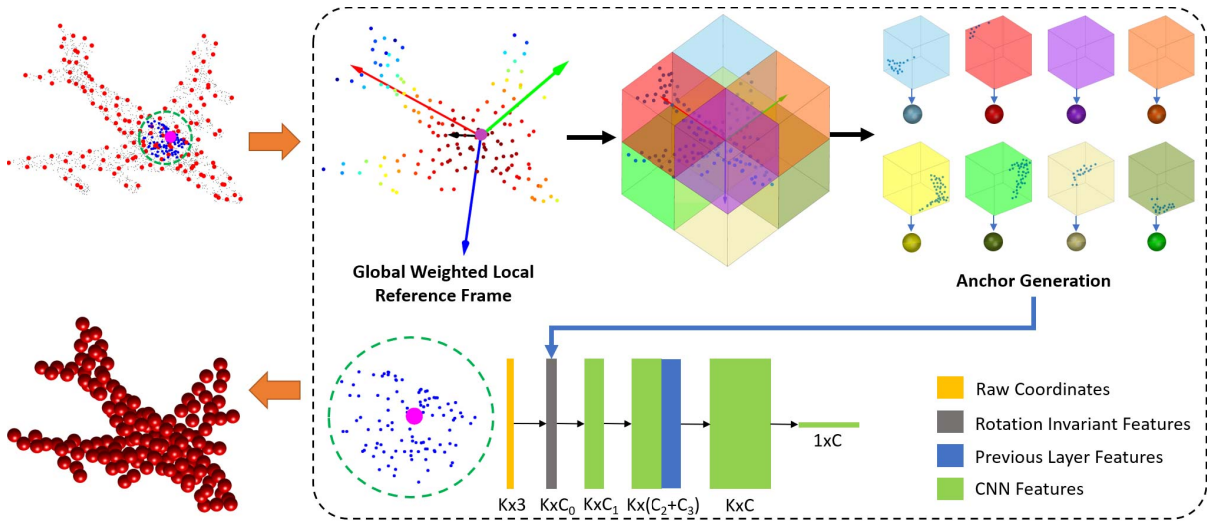

Figure 2. Global context aware convolution (GCAConv) for learning point cloud features comprises of two main steps: (1) Transform into rotation-invariant feature space: for an input point cloud (the upper left plane model), the red dots indicate the keypoints extracted by farthest point sampling. At each keypoint (e.g. the pink dot), we first establish a local reference frame (LRF) by employing weights from *all* other keypoints. The 3D coordinates of the keypoint neighbors are projected to the local space spanned by the LRF to obtain rotation invariant features; and (2) Global feature learning with anchors: eight anchors are constructed to represent eight bins that spans the half-spaces due to the LRF. The local-global relation between the points in a neighborhood and the shape approximates in anchors are folded by a 1D convolution to output final rotation-invariant features.

arise, reducing the distinctiveness of the local features. For example, it is hard to tell apart a corner region on a bed and on a floor/wall/ceiling in the presence of arbitrary rotations. To solve these problems, we establish more reliable LRFs by utilizing all query points of Q in the construction:

$$\Sigma_q = \sum_{i=1}^{N} w_i (q_i - p)(q_i - p)^\top, \tag{2}$$

where w_i is the weight that controls how a point in the point set contributes to the matrix. The weight is defined by

$$w_i = \frac{m - \|q_i - p\|}{\sum_{i=1}^{N} m - \|q_i - p\|}, \tag{3}$$

where $m = \max_{i=1..N}(\|q_i - p\|)$. Intuitively, this weight allows nearby points of p to have large contributions to the covariance matrix, and thus greatly affect the LRF. Points further away from p however can contribute globally to the robustness of the LRF. Such weighted LRF construction is a fundamental step in 3D hand-crafted features [30], which can be easily integrated into our proposed convolution.

A typical problem in defining LRFs is the sign flipping, i.e., the LRF signs should not vary for the same point set [30]. There are multiple ways to resolve the ambiguity; here we disambiguate the signs of the eigenvectors by orienting them to the global vector O defined by

$$O = \sum_{i=1}^{N} w_i (q_i - p), \tag{4}$$

which represents the main orientation of the whole model from the perspective of point p.

4.2. Anchor Point Generation

Theoretically, it is possible to perform convolution on the point set transformed into local coordinates using the constructed LRF. However, it is wasteful to discard global information from the original coordinates as such information can further improve feature distinctiveness. Our idea here is to use anchor points to retain such information in a compact way.

Specifically, to establish the anchors, we divide the whole input point cloud into eight bins, as shown in Figure 2. In each bin, we use the barycenter of the local point set in that bin as the anchor point. Such anchors are crude approximations to the global input shape, and therefore they convey useful information for the convolution.

It is worth noting that there are many ways to define anchors in our case. For example, one can choose to use more bins or all the original point coordinates as anchors, but those will significantly increase computation time for the convolution. We empirically use eight bins as it strikes a balance between the amount of global information retained and the running time.

4.3. Global Context Aware Convolution

With the LRFs and anchors points defined, we are now ready to construct our Global Context Aware Convolution (GCAConv) to learn the rotation invariant features. Let us

consider a point set $P = \{x_i\}$ where x_i represents 3D coordinates of the point i. Let Ω_i be a local point set centered at x_i. A typical convolution to learn the features of Ω_i can be written as

$$\mathbf{f}(\Omega_i) = \sigma(\mathcal{A}(\{\mathcal{T}(\mathbf{f}_{x_i}) : \forall i\})) \tag{5}$$

This formula indicates that features of each point in the point set are first transformed before being aggregated by the aggregation function $\mathcal{A}$ and passed to an activation function σ. A popular choice of $\mathcal{A}$ is maxpooling, which supports permutation invariance in the orders of the input point features [22]. There are a few ways to define the transformation function $\mathcal{T}$. In PointNet [22], it is defined by

$$\mathcal{T}(\mathbf{f}_{x_i}) = \mathbf{w}_i \cdot \mathbf{f}_{x_i} \tag{6}$$

where $\cdot$ indicates the element-wise product, and $\mathbf{w}_i$ is the weight parameter to be learned by the network. This product however ignores the contribution of features from neighboring points x_j to center x_i. To further incorporate such neighbor information, Liu et al. [19] proposed to define the weights by a mapping from a relation vector $\mathbf{h}_{ij}$ between a point x_i and its neighbor x_j.

Here our goal is to define the weights by using the local point set and the anchors. We project both the local point set and anchor points onto the LRF system such that the global 3D coordinates are transformed to a local frame:

$$x'_i = LRF(x_i), \qquad a'_i = LRF(a_i). \tag{7}$$

where x_i and a_i represents the global point and anchor, and x'_i and a'_i represents the local point and anchor, respectively. From here, we aim to relate the weights to such coordinates. Given a pair of a local point x'_i and an anchor a'_j, we define their relation as

$$\mathbf{h}(x'_i, a'_j) = (x'_i - a'_j, \|x'_i - a'_j\|) \tag{8}$$

which can be represented by a 1×4 vector. We stack the features over eight anchors into an 8×4 matrix.

Our convolution can then be defined as a 1D convolution $\mathcal{K}$ that transforms such matrix into a feature vector. The kernel of the convolution is 1×8.

$$\mathcal{T}(\mathbf{f}_{\Omega_i}) = \mathbf{w}_i \cdot \mathbf{f}_{x_i} = (\mathcal{K} \star \mathbf{h_i}) \cdot \mathbf{f}_{x_i} \tag{9}$$

Note that in this formula, we operate on local coordinates, and we use the anchors a'_i to approximate features from neighboring points. This allows us to have two main advantages. First, our convolution only needs local features to operate. Second, the LRFs allow that the learnt features are rotation invariant by definition, without the need of data augmentation during training. Our features can generalize easily to unseen rotations, and we also save a lot of computation during training.

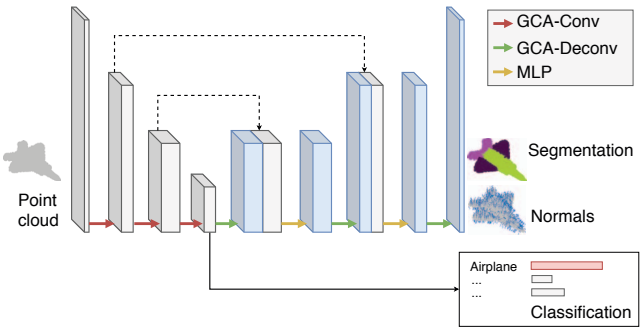

Figure 3. Our network architecture with the proposed point cloud convolution. We use three convolution layers to extract point cloud features before fully connected layers for object classification. We use the same encoder-decoder style architecture with skip connections for object part segmentation and normal estimation task.

4.4. Network Architecture

We use the proposed convolution to design three neural networks for object classification, object part segmentation, and normals estimation, respectively. The architecture is shown in Figure 3. Our classification network has a standard architecture and uses three consecutive layers of convolution (with point downsampling) followed by fully connected layers (256, 128) to output the probability map. In three layers of convolutions, the output channels are set as 128, 256, 512 respectively, and the downsampling numbers are set as 512, 128 and 32 respectively. The neural network for object part segmentation and normal estimation has a decoder branch that includes skip connections and gradually upsamples the point cloud to the original resolution. We use MLP after a skip connection to unify and transform the combined features to have a valid size before deconvolution. Our deconvolution is defined similarly to GCAConv. The minor difference is that it gradually outputs denser points with fewer features.

5. Experimental Results

In this section, we evaluate our method on the 3D object classification, object part segmentation, shape retrieval, and normal estimation task. We implemented our method in TensorFlow [1] with a batch size of 32 to train object classification and 16 to train object part segmentation, shape retrieval, and normal estimation. The training is performed with Adam optimizer with an initial learning rate set to 0.001. The experiments are conducted on a machine with an Intel(R) Core(TM) i7-6900K CPU equipped with an NVIDIA GTX TITAN X GPU.

5.1. Classification on ModelNet40

Object classification is the main task in our evaluation. We train the classification network by using the ModelNet40 variant of the ModelNet dataset [36]. ModelNet40

Method	Format	Input size	Params.	z/z	SO3/SO3	z/SO3	Average acc.	Acc. std.
VoxNet [20]	voxel	30^3	0.9M	83.0	87.3	-	85.2	3.0
SubVolSup [23]	voxel	30^3	17M	88.5	82.7	36.6	69.3	28.4
Spherical CNN [8]	voxel	2×64^2	0.5M	88.9	86.9	78.6	84.8	5.5
MVCNN 80x [28]	view	80×224^2	99M	90.2	86.0	81.5	85.9	4.3
PointNet [22]	xyz	1024×3	3.5M	87.0	80.3	21.6	63.0	41.0
PointNet++ [24]	xyz	1024×3	1.4M	89.3	85.0	28.6	67.6	33.8
PointCNN [17]	xyz	1024×3	0.60M	91.3	84.5	41.2	72.3	27.2
RS-CNN [19]	xyz	1024×3	1.41M	90.3	82.6	48.7	73.9	22.1
RIConv [40]	xyz	1024×3	0.70M	86.5	86.4	86.4	86.4	0.1
SPHNet [21]	xyz	1024×3	2.9M	87.0	87.6	86.6	87.1	0.5
SFCNN [25]	xyz	1024×3	-	**91.4**	**90.1**	84.8	88.8	3.5
ClusterNet [5]	xyz	1024×3	1.4M	87.1	87.1	87.1	87.1	**0.0**
Ours (w/o anchor)	xyz	1024×3	0.21M	86.3	86.2	86.2	86.2	**0.0**
Ours	xyz	1024×3	0.39M	89.0	89.2	**89.1**	**89.1**	**0.0**

Table 1. Comparisons of the classification accuracy (%) on the ModelNet40 dataset. On average, our method has the best accuracy and lowest accuracy deviation in all cases.

contains CAD models from 40 categories such as airplane, bottle, chair, dresser, vase, etc. We use the preprocessed data from PointNet [22] that consists of $9,843$ models for training and $2,468$ models for testing. We use point clouds of size 1024 in this task. Each point is represented by (x, y, z) coordinates in the Euclidean space. The training takes approximately 11 hours to converge in 250 epochs.

Following Esteves et al. [8] and Zhang et al. [40], we evaluate the performance of object classification with three scenarios: (1) using data augmented with rotation about gravity axis (z/z) for training and testing, (2) using data augmented with arbitrary rotations (SO3/SO3) for training and testing, and (3) training with data by z-rotations and testing with data by SO3 rotations (z/SO3). It is expected that rotation-invariant convolutions should work well in the z/SO3 scenario.

Table 1 details the results of this experiment, which confirms the effectiveness of the proposed rotation-invariant convolution. As can be seen, on average, not only our classification accuracy outperforms the state-of-the-art translation-invariant point cloud convolution, the performance is also consistent across three scenarios. For rotation-invariant convolutions, our method outperforms the accuracy of RIConv [40], SPHNet [21], and ClusterNet [5] by a good margin. Our method is slightly more accurate than SFCNN [25] but much more consistent.

5.1.1 Ablation Studies

Network Design. We conduct an ablation study on the ModelNet40 dataset for the classification task (Table 2). We examine four settings in our convolution: (1) the globally weighted LRFs with main orientation (Weight), (2) the use of main orientation to resolve the LRF sign ambiguity (O vector), (3) the use of anchors for global context (Anchor), and (4) the data augmentation with rotations used for the training (Rot. Aug.). Five models (A-E) are used to study the effects of these settings by turning them on/off.

Model A is our baseline setting with all settings on. Model B tests the importance of the weights for computing LRFs and the main orientation. It can be seen that without such weights, the accuracy decreases to 87.1%. The main reason is that the LRFs and the main orientation are more noisy and less repeatable in such case. Next, in model C we further turn off the O vector to test the stability of the LRFs without sign correction. The accuracy further decreases to 86.7%. This verifies that constructing stable LRFs is key to good network performance. In model D, we turn off the global anchor. In this case, only the local points are used for feature extraction. Thanks to the LRFs, the local features are still effective despite of mild accuracy drop. In model E, we test the performance without rotation augmentation scheme during the training procedure. We find the

Model	Weight	O Vector	Anchor	Rot. Aug.	Acc.
A	✓	✓	✓	✓	89.2
B		✓	✓	✓	87.1
C			✓	✓	86.7
D	✓	✓		✓	86.6
E	✓	✓	✓		89.2

Table 2. An evaluation of our network design. It shows that weighted LRF, resolving LRF sign ambiguity, and global anchor play an important role for good performance.

Number of Anchors	1	2	4	8
Accuracy	87.3	87.8	88.5	89.2

Table 3. Classification accuracy (%) on ModelNet40 [36] with different number of anchors.

accuracy is not affected by data augmentation as GCAConv already achieves exact rotation invariance.

Comparison to learned LRFs. It is generally tempting to learn the LRFs to design rotation-invariant convolution. Here we compare this method to our proposed LRFs. We use a two-layer MLP to predict the LRFs and then use them to transform the input point coordinates into a local coordinates before proceeding for convolution as described in the main paper. We found that predicting LRFs works well in z/z and SO3/SO3 mode, with both scenarios achieved accuracies of 89.3% and 89.2%, respectively. However, using data-driven LRFs makes the convolution only *rotation-aware*, but not exactly rotation-invariant. Such convolution fails to generalize to unseen rotations in the z/SO3 scenario with the accuracy of 36.2%.

Number of Anchors. From the ablation studies, we see that without global anchors, the performance is decreased. Here, we further analyze the effects of the number of anchors by investigating the performance on ModelNet40 with a different number of anchors. The qualitative results are shown in Table 3. We can see that with only one anchor, the accuracy decreases to 87.3%, but still higher than RIConv which is around 86.4%. This shows the advantages of global information. With the number goes on, the accuracy also increases. We empirically use eight anchors as it strikes a balance between the amount of global information retained and the running time.

5.2. Object Part Segmentation on ShapeNet

In addition to object classification, we evaluate our method to output a label for each point in the point cloud, resulting in object part segmentation. We use the 3D models in ShapeNet [4] to train our network with point size of 2048 in this task. It takes roughly 36 hours for the training to complete 300 epochs.

The quantitative and qualitative results are shown in Table 4 and Figure 4, respectively. In this task, we achieve start-of-the-art results for both SO3/SO3 and z/SO3 scenarios. Our method outperforms RIConv [40] by almost 2% of accuracy. From Figure 4, we can clearly see that with z/SO3 mode methods like PointNet++ and SpiderCNN can not work well. This is easy to explain as these methods use the raw xyz coordinates as input for training, thus cannot well understand unknown rotations. RIConv [40] works better as it converts xyz coordinates into rotation invariant format like distances and angles before training. However, it still has difficulties in recognizing the boundaries while our method can treat these regions well by incorporating global context information (see column 2 and 3 in Figure 4).

Method	input	SO3/SO3	z/SO3
PointNet [22]	xyz	74.4	37.8
PointNet++ [24]	xyz+normal	76.7	48.2
PointCNN [17]	xyz	71.4	34.7
DGCNN [34]	xyz	73.3	37.4
SpiderCNN [37]	xyz+normal	72.3	42.9
RS-CNN [19]	xyz	72.5	36.5
RIConv [40]	xyz	75.5	75.3
Ours (w/o anchor)	xyz	73.2	73.6
Ours	xyz	**77.3**	**77.2**

Table 4. Comparisons of object part segmentation performed on ShapeNet dataset [4]. The mean per-class IoU (mIoU, %) is used to measure the accuracy under two challenging rotation modes: SO3/SO3 and z/SO3.

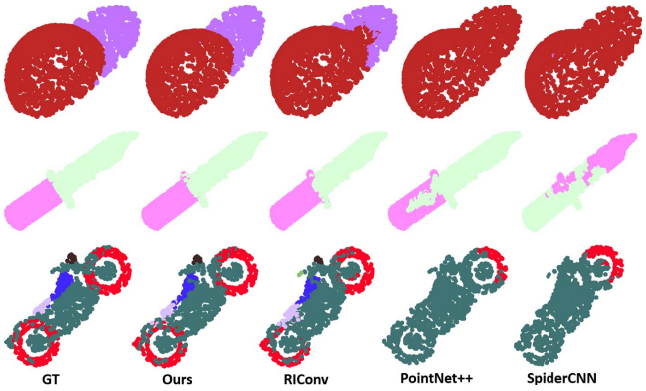

Figure 4. Qualitative comparisons of part segmentation for GCAConv, RIConv [40], PointNet++ [24], SpiderCNN [37] under the z/SO3 rotation mode (from the left column to the right column).

5.3. Shape Retrieval

A popular evaluation of rotation invariance on 3D shape is the shape retrieval task [27]. Here we conducted experiments on ShapeNet Core [36], following the perturbed protocol of the SHREC'17 3D shape retrieval contest [27] and the experiment setting of SFCNN [25]. We use the same output features from the bottleneck layer in the network (similar to features used in the classification task; see Figure 3). We compare with methods proposed in SHREC'17 [10, 29, 3] and two recent methods on rotation-invariant convolution [8, 25]. The results are shown in Table 5. It can be seen that our method achieves the state-of-the-art accuracy, outperforming previous methods for most evaluation metrics.

Method	micro PN	micro R@N	micro F1@N	micro mAP	micro NDCG	macro PN	macro R@N	macro F1@N	macro mAP	macro NDCG	Score
Furuya [10]	81.4	68.3	70.6	65.6	75.4	60.7	53.9	50.3	47.6	56.0	56.6
Tatsuma [29]	70.5	**76.9**	71.9	69.6	78.3	42.4	**56.3**	43.4	41.8	47.9	55.7
Zhou [3]	66.0	65.0	64.3	56.7	70.1	44.3	50.8	43.7	40.6	51.3	48.7
Spherical CNN [8]	71.7	73.7	-	68.5	-	45.0	55.0	-	44.4	-	56.5
SFCNN [25]	77.8	75.1	**75.2**	70.5	**81.3**	65.6	53.9	**53.6**	48.3	58.0	59.4
Ours	**82.9**	76.3	74.8	**70.8**	**81.3**	**66.8**	55.9	51.2	**49.0**	**58.2**	**61.2**

Table 5. Comparisons of 3D shape retrieval on the ShapeNet Core [36]. The accuracy (%) is reported based on the standard evaluation metrics including precision, recall, f-score, mean average precision (mAP) and normalized discounted cumulative gain (NDCG).

Method	z/z	SO3/SO3	z/SO3	Err. std.
PointNet++ [24]	0.34	0.55	0.81	0.24
RS-CNN [19]	**0.26**	0.50	0.83	0.29
RIConv [40]	1.33	1.30	1.30	0.02
Ours	0.42	**0.42**	**0.44**	**0.01**

Table 6. Comparisons of the normal estimation on ModelNet40. The accuracy is reported on three test cases: training and testing with z/z, SO3/SO3 and z/SO3 rotation, respectively. Our method has good accuracy and lowest accuracy deviation in all cases.

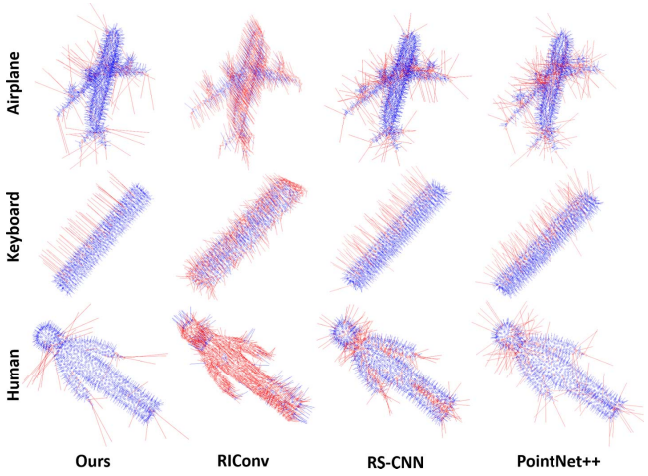

Figure 5. Qualitative comparisons of normal estimation for GCA-Conv, RIConv [40], RS-CNN [19], and PointNet++ [24] under the z/SO3 rotation mode (from the left column to the right column).

5.4. Normals Estimation

Normals estimation for point clouds is instrumental in many applications such as point cloud rendering, feature extraction, and surface reconstruction. Here we conduct normals estimation on point clouds using the ModelNet40 dataset. For each model, we uniformly sample 1024 points from the original data for training. We compute a loss based on the cosines between the predicted unit vectors and the ground truth normals to guide the training. Our results are shown in Table 6.

In this table, our method achieves the best consistency in predicting normals across three test scenarios. In SO3/SO3 and z/SO3 case, our method is the most accurate. It outperforms other methods by a wide margin. The predicted normals are depicted in Figure 5. We quantize the errors by calculating the angles between the predicted and ground truth normals. In Figure 5, the blue and red vectors depict normals with less than 30° and greater than 90° of error. It can be seen that our method is the most accurate visually. It is worth noting that RIConv [40] performs poorly in the normals estimation task because it uses rotation-invariant features that discard the reference coordinate frames, and so the normals of RIConv is not globally consistent.

6. Conclusion

In this work, we introduced a novel approach to design rotation-invariant convolution for 3D point clouds. We show that building robust and repeatable local reference frames is critical to boosting the performance of rotation-invariant object classification. In this task, our newly proposed convolution can match the performance of state-of-the-art translation-invariant convolutions. Our work opens up opportunities to narrow down the performance gap between rotation-invariant and translation-invariant convolution in general 3D deep learning, making robust convolutions for 3D point clouds feasible.

Here we detail a few potential ideas for future research. First, while our method achieves good performance, it is not clear whether local reference frames can be set robustly by a neural network. There is a recent work [42] that attempts to solve this problem, but the performance on object classification needs further investigation. Second, generalizing point cloud convolutions to support non-rigid transformations and deformable objects could further improve overall robustness. Finally, more thorough benchmarking rotation-invariant convolutions with real-world data [31] is necessary to understand the impact of such data on the learning of rotation-invariant features.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016.

[3] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. Jan Latecki. Gift: A real-time and scalable 3d shape search engine. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5023–5032, 2016.

[4] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q.-X. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[5] C. Chen, G. Li, R. Xu, T. Chen, M. Wang, and L. Lin. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4994–5002, 2019.

[6] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Niessner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[7] H. Deng, T. Birdal, and S. Ilic. Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors. In *Proceedings of the European Conference on Computer Vision*, pages 602–618, 2018.

[8] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018.

[9] C. Esteves, Y. Xu, C. Allen-Blanchette, and K. Daniilidis. Equivariant multi-view networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1568–1577, 2019.

[10] T. Furuya and R. Ohbuchi. Deep aggregation of local 3d geometric features for 3d model retrieval. In *BMVC*, volume 7, page 8, 2016.

[11] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[12] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung. Scenenn: A scene meshes dataset with annotations. In *2016 Fourth International Conference on 3D Vision*, pages 92–101, 2016.

[13] B.-S. Hua, M.-K. Tran, and S.-K. Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018.

[14] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[15] R. Klokov and V. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017.

[16] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[17] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In *Advances in neural information processing systems*, pages 820–830, 2018.

[18] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas. Fpnn: Field probing neural networks for 3d data. In *Advances in Neural Information Processing Systems*, pages 307–315, 2016.

[19] Y. Liu, B. Fan, S. Xiang, and C. Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019.

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

[21] A. Poulenard, M.-J. Rakotosaona, Y. Ponty, and M. Ovsjanikov. Effective rotation-invariant point cnn with spherical harmonics kernels. *International Conference on 3D Vision*, 2019.

[22] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[23] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016.

[24] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5105–5114, 2017.

[25] Y. Rao, J. Lu, and J. Zhou. Spherical fractal convolutional neural networks for point cloud recognition. In *Computer Vision and Pattern Recognition*, 2019.

[26] G. Riegler, A. Osman Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017.

[27] M. Savva, F. Yu, H. Su, M. Aono, B. Chen, D. Cohen-Or, W. Deng, H. Su, S. Bai, X. Bai, et al. Shrec16 track: largescale 3d shape retrieval from shapenet core55. In *Proceedings of the eurographics workshop on 3D object retrieval*, volume 10, 2016.

[28] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[29] A. Tatsuma and M. Aono. Multi-fourier spectra descriptor and augmentation with spectral clustering for 3d shape retrieval. *The Visual Computer*, 25(8):785–804, 2009.

[30] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of the European Conference on Computer Vision*, pages 356–369. Springer, 2010.

[31] M. A. Uy, Q.-H. Pham, B.-S. Hua, D. T. Nguyen, and S.-K. Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision*, 2019.

[32] L. van der Maaten and G. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 2008.

[33] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics*, 36(4):1–11, 2017.

[34] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 2019.

[35] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[36] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[37] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision*, pages 87–102, 2018.

[38] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics*, 35(6):1–12, 2016.

[39] Y. You, Y. Lou, Q. Liu, Y.-W. Tai, L. Ma, C. Lu, and W. Wang. Pointwise rotation-invariant network with adaptive sampling and 3d spherical voxel convolution. In *AAAI Conference on Artificial Intelligence*, pages 12717–12724, 2020.

[40] Z. Zhang, B.-S. Hua, D. W. Rosen, and S.-K. Yeung. Rotation invariant convolutions for 3d point clouds deep learning. In *International Conference on 3D Vision*, pages 204–213, 2019.

[41] Z. Zhang, B.-S. Hua, and S.-K. Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1607–1616, 2019.

[42] A. Zhu, J. Yang, C. Zhao, K. Xian, Z. Cao, and X. Li. Lrf-net: Learning local reference frames for 3d local shape description and matching. *Sensors*, 2020.