CONCENTRIC SPHERICAL GNN FOR 3D REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Learning 3D representations that generalize well to arbitrarily oriented inputs is a challenge of practical importance in applications varying from computer vision to physics and chemistry. We propose a novel multi-resolution convolutional architecture for learning over concentric spherical feature maps, of which the single sphere representation is a special case. Our hierarchical architecture is based on alternatively learning to incorporate both intra-sphere and inter-sphere information. We show the applicability of our method for two different types of 3D inputs, mesh objects, which can be regularly sampled, and point clouds, which are irregularly distributed. We also propose an efficient mapping of point clouds to concentric spherical images using radial basis functions, thereby bridging spherical convolutions on grids with general point clouds. We demonstrate the effectiveness of our approach in achieving state-of-the-art performance on 3D classification tasks with rotated data.

1 INTRODUCTION

While convolutional neural networks have been applied to great success to 2D images, extending the same success to geometries in 3D has proven more challenging. A desirable property and challenge in this setting is to learn descriptive representations that are also equivariant to any 3D rotation. Cohen et al. (2018) and Esteves et al. (2018) showed that the spherical domain permits learning such rotationally equivariant representations, by defining convolutions with respect to spherical harmonics. In practice, 3D convolutions are implemented via discretization of the sphere. Earlier spherical CNNs used spherical coordinate grids, but these discretizations result in non-uniform samplings of the sphere, which is non-ideal. Furthermore, spherical convolutions defined on these grids scale with $O(N^{1.5})$ complexity (N as the number of grid points). Subequent works, Jiang et al. (2019), Cohen et al. (2019), Defferrard et al. (2020), designed more scalable $O(N)$ convolutions focusing on more uniform spherical discretizations.

Existing spherical CNNs operate over a spherical image, resulting from projection of data to a bounding sphere. We show that is more expressive and general to instead operate over a concentric, multi-spherical discretization for representing data. Our main innovation is introducing a new two-phase convolutional scheme for learning over a concentric spheres representation, by alternating between inter-sphere and intra-sphere convolutional blocks. We use graph convolutions to incorporate inter-sphere information, and 1D convolutions to incorporate radial information. Similar to Jiang et al. (2019),Cohen et al. (2019), we focus on the icosahedral spherical discretization, which produces a mostly regular sampling over the sphere. Our proposed architecture is hierarchical, following the recursive coarsening hierarchy of the icosahedron. Combinining intra-sphere and inter-sphere convolutions has a conceptual analogy to gradually incorporating information over volumetric sectors. At the same time, our learned representation retains rotational equivariance from scalable graph-based spherical convolutions.

We demonstrate the effectiveness and generality of our approach through two 3D classification experiments with different types of input data: mesh objects and general point clouds. The latter poses an additional challenge for discretizations, as native point clouds are non-uniformly distributed in 3D space.

To summarize our contributions:

1. We propose a novel multi-sphere icosahedral discretization for representation of 3D data, and show that incorporating the radial dimension can greatly enhance representation ability over single-sphere representations.
2. We introduce a novel convolutional architecture for multi-sphere discretization by introducing two different types of convolutions, conceptually separated as intra-sphere and inter-sphere. Combining graph convolutions (intra-sphere) with 1D radial convolutions (inter-sphere) leads to an expressive architecture that is also rotationally equivariant. Our proposed convolutions are also scalable, each being linear with respect to total grid size.
3. We design new mappings for both 3D mesh objects and general point clouds to the proposed representation. We achieve state-of-art performance on ModelNet40 point cloud classification, using the proposed model and a data mapping using radial basis functions. We also improve on existing Spherical CNN performance in SHREC17 3D mesh classificaiton task by utilizing multi-radius information.

2 Related Work

Spherical CNNs. The goal of learning rotationally invariant representations of 3D geometries has led to several ideas for rotationally equivariant convolutions in the spherical domain. Cohen et al. (2018), Esteves et al. (2018) defined spherical convolutions that are rotationally equivariant to rotations of the $SO(3)$ group. However, these convolutions are restricted to non-uniform grid samplings and scale superlinearly with respect to grid resolution. Later works have explored more scalable convolutions on other spherical discretizations, achieving linear complexity with respect to grid resolution.

Jiang et al. (2019) proposed using parameterized differential operators to form convolutional kernels over the icosahedron, where equivariance is restricted to rotations about the z-axis. Cohen et al. (2019) proposed gauge equivariant convolutions on manifolds, operating on feature fields corresponding to underlying geometric entities. This was applied to achieve rotationally equivariant convolutions over the icosahedral discretization. Defferrard et al. (2020) propose a graph convolution-based spherical CNN using spectral filters, along with a distance-weighted nearest-neighbors graph construction scheme that allows balancing between rotational equivariance and efficiency, when applied to different types of grids.

Other spherical CNNs have been designed in the context of handling arbitrary point cloud data, which typically requires first mapping the data to a discretization. Rao et al. (2019) uses graph-convolution inspired message passing oeprators for learning over the icosahedral discretization. Our work is similar to Rao et al. (2019) and Defferrard et al. (2020)) in terms of using graph-based spherical convolutions, but we generalize to multi-sphere convolutions. You et al. (2020) is the most related work in terms of multi-sphere representation learning. The authors propose a spherical voxel grid, and extending the SO3 convolutions of Cohen et al. (2018) to incorporate the radial dimension. Our work treats spherical and radial convolutions as distinct, which observes much better results in practice. We also use more scalable spherical convolutions defined on the uniform icosahedral grid.

Pointwise Convolution Networks. There is a significant body of work on learning point cloud representations using pointwise convolutions, beginning with with Qi et al. (2017) which proposed learning permutation invariant functions that directly operate on point coordinates. Only more recently have such methods have been developed towards learning rotationally invariant representations.

Thomas et al. (2018) and Poulenard et al. (2019) both propose pointwise convolutional filters based on spherical harmonic functions to achieve rotational equivariance (or invariance). Distance information is recorded through learned functions in the former, and radial sampling in the latter. While these filters are defined with respect to all-to-all convolution between points, in practice convolutions are limited to k-nearest neighbors (Poulenard et al. (2019)) for scalability. Chen et al. (2019), Sun et al. (2019), Zhang et al. (2019) all extract rotationally invariant features (i.e. low-level geometric features such as angles and distances) from the point cloud as input to their respective convolutional architectures. These features are hand-engineered based on carefully picking local frames of references, or global in the case of Sun et al. (2019).

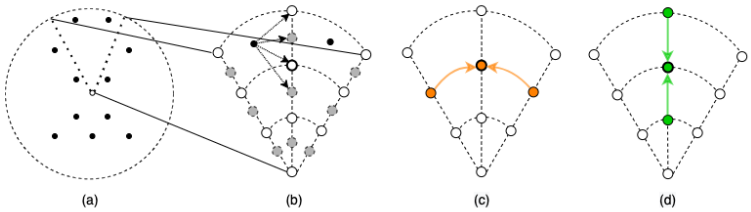


Figure 1: 2D cross-section illustrations of concentric spheres representation. (a) shows an example point cloud with respect to bounding sphere. (b) zooms in on a particular sector. Data points are mapped to RBF values defined on vertices, where an expanded discretization is used. (c) Intra-sphere convolution and (d) inter-sphere convolutions are are applied to the target vertex (bolded). In reality, intra-sphere convolution on local neighborhood of sphere, and involves 5 or 6 vertices.

3 Representation by Concentric Spheres

Existing work on spherical CNNs operate on spherical grids, where data is typically projected to and defined on grid points. However, projecting 3D data to a single sphere may not always be sufficient or appropriate. Simple projections may be lossy when describing highly non-convex shapes, for instance if the shape curves in on itself. To increase capacity to distinguish different data distributions, we introduce a new discretization based on concentric spheres. The introduction of concentric spheres additionally discretizes 3D space in the radial dimension. Single sphere discretization is a special case in our proposed paradigm, corresponding to an outermost bounding radius.

Spherical Discretization. We work with an icosahedral grid discretization of the sphere, as it (1) has a regular (recursive) construction and (2) results in a largely uniform sampling over the sphere. The base icosahedron I_0 has 12 vertices, forming 20 equilateral triangle faces (each face with 3 edges). Each vertex is incident with 5 triangles. It can be recursively refined by subviding each triangle into 4 smaller ones, with the number of vertices scaling by level as $|V| = 10 * 4^l + 2$. See Fig. 2 for icosahedron illustration.

Radial Discretization. We construct a multi-radius discretization for R concentric spheres by stacking stacking R icosahedral grids. The same spherical discretization is shared across concentric spheres, enabling efficient convolutions. Assuming unit radius normalization, we use a uniform discretization that results in concentric spheres scaled to radiuses $[\frac{1}{R}, \frac{2}{R}, ..., 1]$.

Intra-sphere Convolutions.

There is a growing body of work addressing design of rotation-equivariant filters over spherical feature maps. We focus on graph convolutional filters for intra-sphere convolutions, as graph convolutions are scalable and lead to equivariant representations, up to discretization effects (Defferrard et al., 2020). While there is a rich body of work on graph-based convolutions and its variants, this work uses the graph convolution from Kipf & Welling (2017).

This motivates our construction of the undirected graph $G^{(l)} = (V^{(l)}, E^{(l)})$ from a level l icosahedron I_l. Vertices of the vertex set $V^{(l)}$ correspond one-to-one with vertices of I_l projected to unit sphere. $E^{(l)}$ is simply the set of all (bidirectional) face edges of the icosahedron (projected to unit sphere). Each vertex has a degree of either 5 or 6 in this construction. Since each edge corresponds to approximately equal distance between two points of the sphere Wang & Lee (2011), G_l is also treated as an unweighted graph.

Let t index layer, $i \in [0, R-1]$ index the radial dimension, and $u \in [0, V-1]$ index the vertices. Also let Z be parameters, g be feature vectors, and σ indicate nonlinear activation function. The intra-sphere convolution output $h_{i,u}$ is given by Eq. 1, where $N(u)$ indicates adjacent neighbors of vertex u (including u itself).

$$g_{i,u}^{(t+1)} = \sigma\Big(\sum_{v \in N(u)} \frac{1}{\sqrt{d_u d_v}} Z g_{i,v}^{(t)}\Big) \tag{1}$$

The overall complexity of intra-sphere convolution is equivalent to the total grid size $O(N)$, where $N = R * V^{(l)}$. In practice, R can also be restricted to a relatively small constant.

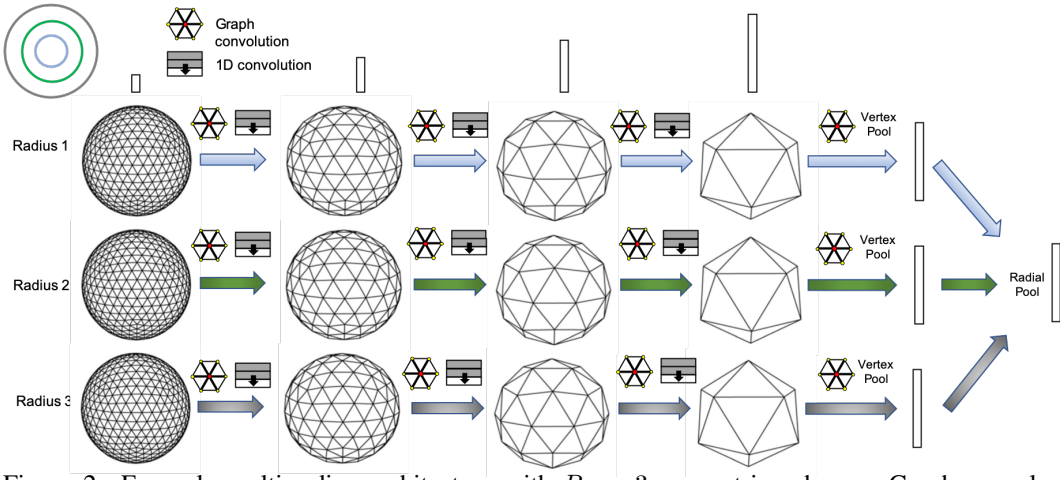


Figure 2: Example multi-radius architecture with $R = 3$ concentric spheres. Graph convolutions, followed by radial convolutions, are applied over a sequence of discretization levels. Pooling coarsens the discretization to a lower level. Vertex-wise and radial-wise pooling is applied to obtain a final representation for classifier. Icosahedron visualization from Satoh et al. (2014).

Inter-sphere Convolutions. We introduce *radial convolutions* to incorporate inter-sphere information, implemented as the 1D convolutions over multiple radial representations of each vertex. Importantly, radial convolutions are also rotationally invariant, as 1D convolution operates over channels of the same vertex. See Fig. 1 for illustration of radial and graph convolutions with respect to concentric spheres representation.

We introduce additional notation to describe radial convolutions. Let Z be parameters, and h feature vectors. Then the inter-sphere convolution output $g_{i,u}$ convolution is given by Eq. 2. K indicates 1D kernel size. We assume K to be odd valued, and pad inputs in the radial dimension to ensure dimension R is maintained across convolutions.

$$h_{i,u}^{(t+1)} = \sigma\Big(\sum_{k=-\lfloor \frac{K}{2} \rfloor}^{\lfloor \frac{K}{2} \rfloor} W_{k+\lfloor \frac{K}{2} \rfloor} h_{i+k,u}^{(t)}\Big) \tag{2}$$

The overall complexity of intra-sphere convolution is also $O(N)$. Intra-sphere convolutions require Z parameters and inter-sphere convolutions require $K * W$ parameters.

Concentric Spherical GNN Architecture. Fig. 2 gives an example illustration of an end-to-end architecture using both convolutions. Importantly, radial convolution blocks are introduced alongside graph convolutional blocks at every level of the spherical discretization hierarchy, to incorporate inter-sphere information gradually. From a icosahedron of level L refinement, we construct a sequences of graphs [G_L, G_{L-1}, ..., G_0]. Each G_l carries an additional R dimension, corresponding to spheres at different radial levels. Each level l features two blocks of convolutions: graph convolutions, followed by radial convolution. These correspond to intra-sphere and inter-sphere convolutions respectively. This is followed by vertex neighborhood pooling, which downsamples the graph from G_l to G_{l-1}. The size of the radial dimension remains constant, until final pooling.

4 Point Cloud to Concentric Spherical Signal

We consider the problem of mapping a point cloud $P \in \mathbb{R}^{N\times 3}$ point cloud to an initial spherical feature map $M \in \mathbb{R}^{R\times V\times F}$, where F is number of input channels. While the concentric grid representation is defined discretely at fixed positions, the space of data point locations is continuous. We also aim to capture the distribution of points in a continuous way. To do so, we summarize the contribution of points using the Gaussian radial basis function:

$$f(x) = \sum_{j=1}^{N} \phi(||x - p_j||_2^2) \tag{3}$$

Here N is the number of data points, and $\phi = \exp(-\gamma r^2)$, parameterized by the bandwidth γ. In practice we limit computation to a local neighborhood (instead of considering all points), and choose γ accordingly. See Sec. A.1 for additional details.

One possible mapping is to compute Eq. 3 at every vertex position $x_{i,u} : i \in [0, R-1], u \in V$ of the spherical discretization, resulting in a single channel feature map. However, it is possible to obtain better resolution in capturing distribution of surrounding points by further sub-diving the discretized space (taking inspiration from Meng et al. (2019)) along the radial dimension. Subdividing along radial dimension by a factor K_e results in a new spherical discretization with size $R' = R * K_e$ in the radial dimension. We compute the RBF value at every vertex position of this new discretization, resulting in a feature map of dimension of $M' \in \mathbb{R}^{R' \times V \times 1}$. We then map back to the original discretization by assigning $M_{i,u} = [M'_{j,u} : j \in (iK_e, iK_e + 1, ..., 2iK_e, 2iK_e + 1, ..., 3iK_e - 1)]$, resulting in a $[R \times V \times 2 * K_e]$ feature map. In other words, to better capture data distribution, multiple RBF values are assigned to each vertex by further sub-dividing space in the radial dimension.

5 EXPERIMENTS

5.1 MODELNET40 POINT CLOUD CLASSIFICATION

We consider the ModelNet40 3D shape classification task, with 12308 shapes and 40 classes. Following convention, each point cloud has 1024 points. For all experiments, 9840 shapes are used for training and 2468 for testing.

Architecture and Hyperparameters
Fig. 3 shows a complete architecture overview. Point clouds are first mapped to 16 concentric spheres with level 4 icosahedral discretization ($L = 4, R = 16$), using RBF kernels with threshold $T = 0.01$. 1D convolutions use a kernel size of 3. Each graph and 1D convolution is followed by batch normalization and ReLU as nonlinear activation. Additionally, skip connections are added between every graph convolution layer, whenever input dimension matches output. The model is trained with Adam optimizer for 30 epochs using initial learning rate of 1e-3, along with learning rate decay by 0.1 at 15 and 25 epochs. The batch size is 32.

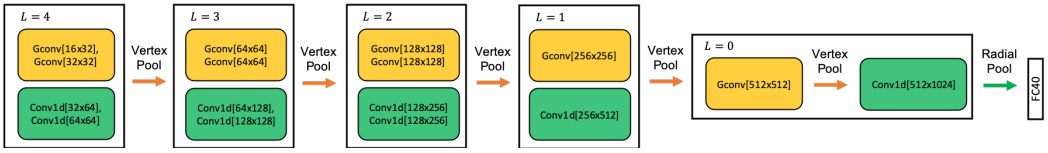


Figure 3: Architecture for ModelNet40 classification. Input dimension is 16, resulting from point cloud RBF data mapping. "Gconv" is graph convolution applied over graph connectivity of the sphere. "Conv1d" is 1D convolution, applied over the radial dimension. L denotes discretization level, as the representation is coarsened following each vertex pooling step. A final pooling of radial dimension results in a 1024 dimensional vector.

Results

We present our results and compare against other related works in Table 1, in four different train/test data orientation settings. When training with rotations, a new rotation is sampled per instance in each epoch. Rotation is the only augmentation used in comparisons. We report accuracy as average validation score across last 5 epochs of training, due to lack of standard validation/test split and to account for variation.

Our method achieves state-of-the-art results in z/SO3 and SO3/SO3 settings, i.e. testing on arbitrarily rotated data. For more detailed comparison, we loosely categorize compared works by method into two categories: pointwise convolution networks and spherical CNNs. Methods in the former

[1]Using numbers reported in respective authors' paper.

[2]ClusterNet, RIConv, and SRINet are designed to be strictly rotationally invariant. Nearly identical performance is expected across all settings (including settings not reported by respective authors).

Table 1: ModelNet40 classification results, across four train/test data orientation settings. NR denotes original data (no rotations), z is arbitrary rotation about z axis, and $SO3$ is arbitrary rotation. For example, SO3/SO3 means training and testing with arbitrary rotations of the data.

Method	Input	z/z	z/SO3	NR/SO3	SO3/SO3
Pointwise Convolution					
PointNet Qi et al. (2017)	xyz	0.875	0.229	0.081	0.849
ClusterNet Chen et al. (2019)	xyz	0.871[1]	0.871[1]	*[2]	0.871[1]
RIConv Zhang et al. (2019)	xyz	0.865[1]	0.864[1]	*[2]	0.864[1]
SPHNet Poulenard et al. (2019)	xyz	0.865	0.856	0.854	0.870
SRINet Sun et al. (2019)	xyz+normal	*[2]	*[2]	**0.870**[1]	*[2]
Spherical CNN					
SFCNN Rao et al. (2019)	xyz	**0.888**	0.831	0.350	0.874
PRIN You et al. (2020)	xyz	0.819	0.765	0.753	0.810
Ours (CSGNN)	xyz	0.884	**0.874**	0.833	**0.884**

category operate directly on data points in 3D space, while methods in the latter operate on a spherical discretization. Our work is most closely related to methods in the spherical CNN category.

Similar to our work, PRIN also explored learning a concentric spherical representation based on extending SO3 convolutions from Cohen et al. (2018). Our method is based on separate graph and radial convolutions, which achieves much better performance in all settings. SFCNN has similarity to our work using graph convolution-inspired message passing filters, and hierarchically learning over the icosahedral discretization of the sphere. However, SFCNN is restricted to a single-sphere representation, and also relies on a PointNet-like learned module to project points to spherical features. While this learned projection should be able capture some degree of multi-radius information in the point distribution, best results seem to be achieved by learning from both intra-sphere and inter-sphere convolutions. Compared to our approach, SFCNN also has a relatively higher performance gap between z/z and any SO3 test setting (significantly so for z/SO3 and NR/SO3), which suggests greater difficulty achieving rotational invariance. CSGNN, similar to SFCNN and PRIN, exhibits some drop in performance in the z/SO3 and NR/SO3 settings. This could partly be due to effects of discretization and data mapping. However, this gap is relatively smallest in CSGNN, and there is essentially no gap in z/z and SO3/SO3 performance.

ClusterNet, RIConv, and SRINet use hand-crafted rotationally invariant geometric features as inputs, and so there is negligble to no performance gap in testing with or without rotations. By contrast, our method largely learns to extract features directly from the input (outside of an initial step mapping points to vertices). PointNet, unlike the other baselines, was not designed to be rotationally equivariant. This reflects in the relatively significant difference in test performance with and without rotations. Even when training with rotations, and using a learned alignment module that attempts to learn a canonical transformation, SO3/SO3 performance is not competitive with that of most other baselines.

5.2 SHREC17 3D SHAPES

The SCHREC17 task has 51300 3D models and 55 categories. We use the version where all models have been randomly perturbed by rotations. Here the inputs are not point clouds, but mesh objects. Cohen et al. (2018),Esteves et al. (2018) presented a ray-casting scheme to regularly sample information incident to outermost mesh surfaces and obtain features maps defined over the spherical discretization. For sufficiently non-convex mesh objects, a single sphere projection may result in information loss. For example, when a ray is incident to multiple surfaces occurring at different radii, this information is discarded by existing methods. We propose a new data mapping that generalizes single sphere representation to a concentric spherical representation, thereby preserving more information. Fig. 4 in appendix shows visual examples of where the proposed representation may be helping.

Representation
Conceptually, we perform ray-casting from multiple concentric spheres towards the mesh center, where each sphere has been scaled to different radii. Recording 1st hits from each sphere results in

a multi-radius rojection, where we use a uniform $[\frac{1}{R}, \frac{2}{R}, ..., 1]$ radii division assuming normalized inputs. We record distance (with respect to outermost sphere), sin, and cos features from each ray intersection similar to in other related work (but do not use convex hull information), resulting in 3 features per vertex.

Method	F_1
Cohen et al. (2018) (equiangular, $b = 64$)	0.789[3]
Esteves et al. (2018) (equiangular, $b = 64$)	0.794[3]
DeepSphere Defferrard et al. (2020) (equiangular, $b = 64$)	0.794[3]
DeepSphere Defferrard et al. (2020) (HEALPix, $N_{side} = 32$)	0.807[3]
CSGNN (icosahedral, $L = 4, R = 1$)	0.805
CSGNN (icosahedral, $L = 4, R = 16$)	0.823

Table 2: SHREC17 classification performance in terms of F_1 metric (micro-average). CSGNN is our implementation. Equiangular, HEALPix, and icosahedral are different discretizations of the sphere. CSGNN (this work) uses level 4 icosahedral discretization, R is number of concentric spheres (specific to this work).

Architecture and Hyperparameters
The architecture for SHREC17 is identical to the one used for ModelNet40 in Fig. 3, with the exception that the input dimension is 3 (corresponding to features obtained from ray-casting). We consider two model variations, single-sphere ($R = 1$) and multi-sphere ($R = 16$). For $R = 16$, we use a 1D convolution kernel of size 3. For $R = 1$, we use a 1D convolution kernel of size 1. Note this is equivalent to applying FC layers; we found adding additional FC layers after graph convolutions helped improve performance in the single-sphere case. Training settings are also same as for ModelNet40, except learning rate decay is at epochs 10 and 20.

Results
See Table 2 for classification results. The reported metric is F_1 micro-average classification score. Results from three other Spherical CNN works designed for general rotational equivariance are included for reference. DeepSphere is most similiar to our work in terms of graph-based spherical convolutions, where the authors explored design of rotationally-equivariant graph convolutional filters with respect to the type of grid and neighborhood size. This work focuses graph construction to the icosahedral discretization, using a minimal set of roughly equidistant neighbors.

We additionally introduce inter-sphere convolutions with concentric spheres, which is largely orthogonal to the design of intra-sphere representation and convolutions. Single-sphere ($R = 1$) CSGNN, a special case of this this work, seems to achieve competitive performance with other single-sphere baselines. However, it is difficult to draw comparative conclusions in this particular case, due to differences in feature extraction, spherical discretization type and size, and model size. More significantly, using multiple spheres ($R = 16$) achieves 2.2% relative performance improvement over the $R = 1$ version. It also seems likely that the concentric spheres approach can be adapted to other spherical convolutional designs as well, but this is beyond the scope of this work.

5.3 Ablation Study

To study the impact of multi-radius spherical discretization, we vary the number of radial levels and present results in Table 3. ModelNet40 is used for all ablation experiments. We also use a base model with $R = 16$ and $L = 4$, and keep the number of parameters identical across all cases. For this particular ablation, we use a single channel, indicator feature map–a special case of the RBF mapping where $\gamma = 0$ and $F = 1$. This eliminates γ as a tuning parameter when varying R. Adding radial convolutions in the case of $R = 1$ is equivalent to adding additional dense layers after graph convolutions. We use the same architecture in 3, except input dimension is 1. Performance consistently improves with higher radial dimension, peaking at $R = 16$ with 4.8% relative accuracy improvement over the $R = 1$ version. Performance declines for $R = 32$, which suggests diminishing returns (as model capacity may need to be increased).

[3]Numbers as reported in Defferrard et al. (2020).

Setting	$R = 1$	$R = 4$	$R = 8$	$R = 16$	$R = 32$
SO3/SO3	0.839	0.857	0.869	0.879	0.872

Table 3: ModelNet40 ablation with number of radial levels (R). Total number of parameters is fixed across all settings.

More ablation studies are presented in Table 4. We study the impact of varying the radial kernel size $K_{RC} = [1, 3, 5]$. $K_{RC} = 1$ is same as learning representations independently learned at each radial level. While this still improves over single-sphere representation, using spatial filters ($K_{RC} = [3, 5]$) over the radial dimension is important for best performance. Varying the number of graph and radial convolutional layers per block shows using between 1 and 2 layers per block led to comparable performance. Finally, we compare using either graph convolutions or radial convolutions. Results suggest that it is essential to combine both types of convolutions for best performance. Interestingly, restricting to radial convolutions achieves slightly better performance than restricting to graph convolutions over the single sphere. This provides further empirical support for the expressiveness of our proposed representation and radial convolutions.

Setting	SO3/SO3	Setting	SO3/SO3
Radial kernel size		*Convolution layers*	
$K_{RC} = 1$, $M_{GC} = 1, M_{RC} = 1$	0.853	$K_{RC} = 3$, $M_{GC} = 1, M_{RC} = 1$	0.880
$K_{RC} = 3$, $M_{GC} = 1, M_{RC} = 1$	0.880	$K_{RC} = 3$, $M_{GC} = 2, M_{RC} = 2$	0.876
$K_{RC} = 5$, $M_{GC} = 1, M_{RC} = 1$	0.882		
Graph convolution only		*Radial convolution only*	
$R = 1$, $M_{GC} = 1$	0.837	$K_{RC} = 3$, $M_{RC} = 1$	0.845

Table 4: Ablation study on ModelNet40. K_{RC} is size of radial convolutional kernel, M_{GC} and M_{RC} are number of graph and radial convolutional layers per block. $R = 16$ and $L = 4$, unless stated otherwise.

6 Discussion and Conclusions

In this work we proposed a new multi-sphere convolutional architecture, CSGNN, for learning rotationally invariant representations of 3D data. We introduced distinct intra-sphere and inter-sphere convolutions, which can be combined to learn more expressive representations compared to being restricted to single-sphere representation. Our use of graph and 1D convolutions preserves rotational equivariance, while achieving linear scalability with respect to size of discretization. We achieve state-of-the-art performance in ModelNet classification for testing on arbitrary rotations among both spherical CNN and pointwise convolutional models. We also show that our approach generalizes to classification of 3D mesh obejcts, by improving on single-sphere representations for the SHREC17 task.

One avenue of future work is to explore more descriptive mappings of point cloud data to the discretization. A learned assignment may better learn vertex features for describing nearby points, for instance. There is also room to explore other kinds of convolutions for incorporating inter-sphere information, as well as other radial division schemes. Finally, existing implementations in this work can be more efficient implementations based on using regular properties of the icosahedral grid, as opposed to using a general graph construction.

References

Chao Chen, Guanbin Li, Ruijia Xu, Tianshui Chen, Meng Wang, and Liang 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*, pp. 4994–5002, 2019.

Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In *International Conference on Machine Learning*, pp. 1321–1330, 2019.

Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. *International Conference on Learning Representations*, 2018.

Michal Defferrard, Martino Milani, Frdrick Gusset, and Nathanal Perraudin. Deepsphere: a graph-based spherical cnn. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=B1e3OlStPB`.

Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning SO(3) equivariant representations with spherical CNNs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 52–68, 2018.

Chiyu Max Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Niessner. Spherical CNNs on unstructured grids. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkl-43C9FQ`.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017.

Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. Vv-net: Voxel vae net with group convolutions for point cloud segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8500–8508, 2019.

Adrien Poulenard, Marie-Julie Rakotosaona, Yann Ponty, and Maks Ovsjanikov. Effective rotation-invariant point cnn with spherical harmonics kernels. In *2019 International Conference on 3D Vision (3DV)*, pp. 47–56. IEEE, 2019.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, pp. 652–660, 2017.

Yongming Rao, Jiwen Lu, and Jie Zhou. Spherical fractal convolutional neural networks for point cloud recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 452–460, 2019.

Masaki Satoh, Hirofumi Tomita, Hisashi Yashiro, Hiroaki Miura, Chihiro Kodama, Tatsuya Seiki, Akira T Noda, Yohei Yamada, Daisuke Goto, Masahiro Sawada, et al. The non-hydrostatic icosahedral atmospheric model: Description and development. *Progress in Earth and Planetary Science*, 1(1):18, 2014.

Xiao Sun, Zhouhui Lian, and Jianguo Xiao. Srinet: Learning strictly rotation-invariant representations for point cloud classification and segmentation. In *Proceedings of the 27th ACM International Conference on Multimedia*, pp. 980–988, 2019.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Ning Wang and Jin-Luen Lee. Geometric properties of the icosahedral-hexagonal grid on the two-sphere. *SIAM Journal on Scientific Computing*, 33(5):2536–2559, 2011.

Yang You, Yujing Lou, Qi Liu, Yu-Wing Tai, Weiming Wang, Lizhuang Ma, and Cewu Lu. Pointwise rotation-invariant network with adaptive sampling and 3D spherical voxel convolution. *The AAAI Conference on Artificial Intelligence*, 2020.

Zhiyuan Zhang, Binh-Son Hua, David W Rosen, and Sai-Kit Yeung. Rotation invariant convolutions for 3d point clouds deep learning. In *2019 International Conference on 3D Vision (3DV)*, pp. 204–213. IEEE, 2019.

A APPENDIX

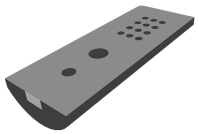
A.1 POINT CLOUD TO SPHERICAL SIGNAL

Instead of for each vertex summing over RBF values with respect to all points, for each point we update the features of vertices in its local neighborhood. This is more efficient when the number of points is less than the number of vertices. Ignoring boundary conditions and degenerate cases, any given point p is contained within two bounding "triangles" of the discretization, corresponding to vertices $(u^{(i)}, v^{(i)}, w^{(i)})$ and $(u^{(i+1)}, v^{(i+1)}, w^{(i+1)})$, where i indicates radial level. We define this as the local neighborhood of p. In this case, we can compute Eq. 3 with respect to each neighboring vertex and update accordingly.

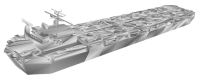
However, using a single γ value for the Gaussian RBF would result in scaling inconsistency: distances between vertices progressively shrink as we move towards inner spheres. Based on the assumption that RBF values should be invariant to scale, we determine a different γ and corresponding RBF depending on radial level. This is determined by first computing the maximum distance d_{max} between any 6 vertices of the two bounding triangles, and using d_{max} to select γ. Specifically, we pick γ such that $\gamma = \frac{-\log T}{d_{max}^2}$, where T is a lower bound target RBF value (tuning parameter). For example, $T = 1$ would correspond to $\gamma = 0$, or a RBF value of 1 for any distance.

Note that distances between the bounding vertices $(u^{(i)}, v^{(i)}, w^{(i)})$ and $(u^{(i+1)}, v^{(i+1)}, w^{(i+1)})$ are approximately equal across the same radius level, and so we consistently use one as reference for computing γ.

A.2 SHREC17 VISUALIZATION

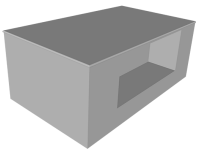


(a) Remote



(b) Watercraft

(c) Cabinet



(d) Table

(e) Lamp

(f) Tower

(g) Dish washer

(h) File cabinet

Figure 4: SHREC17 mis-predicted class pairs from single-sphere model, where multi-sphere model showed biggest relative improvement. Each image is a representative sample from the class. Note that *watercraft*, *table*, and *tower* all have more non-convex features that distinguish them from their mis-predicted counterparts. The concentric spherical model seems to better capture these differences.