RIMeshGNN: A Rotation-Invariant Graph Neural Network for Mesh Classification

Bahareh Shakibajahromi, Edward Kim, David E. Breen
Department of Computer Science, Drexel University
Philadelphia, PA, USA
bs643@drexel.edu, ek826@drexel.edu, deb39@drexel.edu

Abstract

Shape analysis tasks, including mesh classification, segmentation, and retrieval demonstrate symmetries in Euclidean space and should be invariant to geometric transformations such as rotation and translation. However, existing methods in mesh analysis often rely on extensive data augmentation and more complex analysis models to handle 3D rotations. Despite these efforts, rotation invariance is not guaranteed, which can significantly reduce accuracy when test samples undergo arbitrary rotations, because the analysis method struggles to generalize to the unknown orientations of the test samples. To address these challenges, our work presents a novel approach that employs graph neural networks (GNNs) to analyze mesh-structured data. Our proposed GNN layer, aggregation function, and local pooling layer are equivariant to the rotation, reflection and translation of 3D shapes, making them suitable building blocks for our proposed rotation-invariant network for the classification of mesh models. Therefore, our proposed approach does not need rotation augmentation, and we can maintain accuracy even when test samples undergo arbitrary rotations. Extensive experiments on various datasets demonstrate that our methods achieve state-of-the-art performance.

1. Introduction

With the advancements in 3D scanning technology and modeling software, the amount of generated 3D geometric data has increased exponentially. These data are used in various fields such as gaming, augmented and virtual reality, medical imaging, robotics, self-driving cars, and many others. The analysis of 3D mesh data is challenging, and traditional methods can be time-consuming and limited in their capabilities. 3D shape classification and 3D shape matching are among the main applications of shape analysis and have been a long standing topic of active research. These tasks aim to categorize 3D shapes into different classes based on their geometric features and retrieve similar shapes based on user queries. A descriptive and versatile representation of 3D object shapes is crucial for accomplishing these tasks. In the field of 3D shape analysis, deep learning methods such as PointNet [40], PointNet++ [42], ShapeContextNet [57], PointGrid [26], DynamicGCN [54], and SampleNet [25] have demonstrated potential. In the context of 3D shape analysis, it is crucial for a neural network-based model to be able to effectively learn relevant latent features, even when the 3D shape has undergone geometric transformation.

In this study, we employ adaptively sampled mesh models to represent 3D shapes, instead of other popular methods, such as volumetric grids (voxels) [33, 41] and point clouds [42,59]. Mesh models may provide higher resolution by assigning more vertices and polygons to complex geometric areas. Conversely, in extensive, nearly flat regions only a minimal number of polygons are utilized. Mesh models are a useful choice due to their conceptual simplicity, versatility, and efficient processing methods.

Shape analysis tasks, such as mesh classification, segmentation, and retrieval exhibit symmetries in Euclidean space. These tasks involve predicting outcomes that should be unaffected by rotation, translation, and reflection, collectively referred to as $E(3)$ transformations, that are applied to the analyzed geometric models. However, the majority of the mesh [3,21,24], point cloud [18,29,42,54,55], volumetric grids [33,37] analysis methods are not robust to rotation. Most 3D shape analysis methods have been trained and evaluated on 3D shape datasets, such as ModelNet [56] and ShapeNet [7], which contain shapes that have been prealigned into a standard orientation. These methods fail to generalize to models in unknown poses and experience a significant performance drop when evaluated on objects in arbitrary orientations. A different set of techniques deals with the input mesh as a graph and uses graph deep learning techniques to analyze the input mesh for various tasks such as classification and segmentation [34,35,44,52]. However, most spatial GNN layers, such as graph convolution, graph

attention, and message passing layers, are not designed to handle node features that include attributes that vary with different coordinate systems. This means that the spatial coordinates and normal vectors of nodes change when subjected to translation, rotation, and reflection. Consequently, training a mesh classification network using these GNN layers is expensive and requires significant data augmentation to expose the network to 3D meshes in different orientations. Additionally, these GNN layers lack translation/rotation equivariance or invariance, requiring $O(\delta^{-3})$ more filters to achieve an angular resolution of δ (which is assumed to be less than 1) [50]. To tackle these challenges and reduce the computational complexity of the classification network, we propose an approach that is translation, rotation and reflection invariant.

The objective of our research is to develop a representation of 3D objects that can be used for 3D shape classification that remains unaffected by geometric transformations such as rotation and translation. To achieve this goal, we construct a graph neural network framework with mesh-specific features. We propose a unique Graph Neural Network (GNN) as an instance of the message-passing GNN that can jointly learn node, edge, and graph embeddings. Our rotation-, reflection- and translation-equivariant layer uses geometric features of the mesh that change under rotation and translation, such as position and normals, as well as rotation and translation invariant features. Furthermore, we introduce a novel permutation and rotation invariant aggregation function as a component of the proposed GNN. This function leverages the coordinates of vertices by binning them into spherical regions, which are then represented by a complete graph. Each node in the complete graph represents a bin and is assigned a feature vector that is calculated by taking the average of the rotation-invariant features of mesh vertices located in the corresponding spherical shell. We then utilize two layers of Graph Attention Networks (GATs), which employ self-attention over the node features, giving varying importance to different neighbors. We aggregate the node features of the mesh by taking the average of the embedded features of the nodes of the complete graph. Finally, we present a local pooling layer, which implements mesh coarsening at different levels, enabling multi-resolution mesh analysis. Combined together, these components create RIMeshGNN, a (R)otation-(I)nvariant graph neural network (GNN) for mesh classification.

The rest of this paper is structured as follows. In Section 2 we provide a summary of related research. Section 3 introduces the fundamental components of our approach and the framework implemented for 3D mesh classification. Next, in Section 4 we discuss the experimental evaluation of our proposed model for 3D mesh classification. We also provide a comparison to relevant supervised models for 3D classification. We conclude the paper in Section 5 and outline future work. In order to demonstrate the impact and the individual contribution of each component within our classification network, we conducted a series of ablation studies. The outcomes of these studies can be found in the supplementary materials.

2. Related Work

A triangular mesh, composed of vertices (V), edges (E), and faces (F), is the most widely used and effective 3D shape representation in computer graphics. However, the irregularity and unstructured nature of this representation poses a challenge when applying Convolutional Neural Networks (CNNs) to it, as each vertex has a varying number of neighbors at differing distances. Traditional deep learning techniques tackle the irregularity of meshes by either converting them into volumetric grids [33, 37, 41] or creating multiple 2D projections [23, 47] to make them compatible with CNNs. An alternative method involves extracting 3D point samples from the surface of a three-dimensional object and then applying a point cloud technique to the samples [40, 42, 48, 49, 53, 60]. These techniques aim to learn a representation for each point by analyzing its neighbors, using either multi-layer perceptrons or convolutional layers.

Recently, there has been a growing research trend addressing 3D meshes directly. Early work on learning representations for 3D meshes generalized standard 2D Convolutional Neural Networks (CNNs) to 3D. These generalizations treat 3D meshes as manifolds and apply patch-based techniques on the underlying surface [31, 32, 35]. However, it is worth mentioning that most patch-based methods entail high computational complexity, hand-crafted techniques, and precomputed local systems of coordinates [4].

Researchers have attempted to establish a specific order for vertex neighborhoods to prevent expensive resamplings [3, 4, 17, 24, 31]. Lim *et al*. [31] and Gong *et al*. [17] have introduced sampling operators that serialize vertex neighborhoods along spiral trajectories to capture local geometric structures. An alternative method for handling the irregular structure of 3D meshes utilizes several random walks to traverse the surface of the mesh, thereby examining its local and global geometry [3, 24]. These walks are then fed into a Recurrent Neural Network (RNN) that can retain the history of each walk. To overcome the limitations of fixed discretization schemes, DiffusionNet [45] utilizes heat propagation across the mesh surface, generating a diffusion kernel that encodes vertex relationships in terms of connectivity and proximity. Point cloud methods tackle challenges arising from the irregular domain and the absence of ordering in various ways [18, 42, 54, 55]. PCT [18] leverages the permutation invariant properties of transformers, combined with farthest point sampling and nearest neighbor search, to facilitate efficient point cloud learning. DeltaConv [55] adopts the DGCNN network architecture [54] while substi-

tuting each EdgeConv block with an anisotropic convolution layer.

Another way to tackle the irregular, non-uniform structure of a mesh is to exploit the inherent properties of mesh elements that have regular forms. Particularly, in a 2-manifold triangle mesh each edge is incident to exactly two triangular faces and, therefore, adjacent to four other edges. MeshCNN [20] designed a convolution operator for edges, where its spatial support is defined using the four incident edges. Similarly, face-based techniques take advantage of the fact that each face in a 2-manifold triangle mesh is adjacent to exactly three other faces. MeshNet [14] leverages two mesh convolutional layers to aggregate the 1-ring neighbors and learn the spatial and structural features of a face. SubdivNet [21] has also introduced a convolutional operation for mesh faces that supports kernel size, stride, and dilation. However, this method requires a mesh with a subdivision sequence.

As triangle meshes are structured graphs, learning algorithms inspired by graph-based approaches can often be employed to learn from meshes. Our method is closely related to a family of approaches that utilizes the graph representation of 3D meshes and relies on Graph Neural Networks (GNNs). GNNs extract neighborhood features through a weight-sharing strategy and are classified into two main categories: 1) spectral- and 2) spatial-based methods. The spectral-based methods use the eigen decomposition of the Laplacian associated with the graph [5, 9, 27]. Such models generally suffer from large computational complexity, unstable decomposition, and poor generalization across different graphs [1, 35, 52]. In contrast, spatial-based methods aggregate deep feature information of neighboring nodes [19, 36, 51]. These methods have attained considerable attention due to their low computational complexity, generalization, and localized properties. Milano *et al*. [34] proposed PD-MeshNet that performs dynamic, attention-based feature aggregation using the primal-dual graph convolutional framework of Monti *et al*. [36]. Shakibajahromi *et al*. [44] introduced a 3D segmentation network, HyNet, that converts a 3D mesh into a hybrid graph consisting of three types of nodes representing vertices, edges, and faces.

Nevertheless, most end-to-end deep neural networks discussed earlier do not demonstrate invariance to 3D rotations. As a result, the accuracy of these networks tends to decline when the orientations of the input 3D objects differ between training and inference. Several studies have reported a substantial performance difference in shape analysis tasks between aligned and unaligned or rotationally augmented settings, such as those by Poulenard *et al*. [39], Tao *et al*. [49] and Sun *et al*. [48]. Recently several point cloud techniques have been offered to address this issue [10, 28, 39, 48, 49, 61]. For example, Deng *et al*. [10] extended neurons from 1D scalars to 3D vectors, introducing SO(3)-equivariant neural networks for point-cloud processing. Li *et al*. [28] convert the point clouds into their canonical poses by developing a pose selector module that disambiguates PCA-based canonical poses. To address the same issue for meshes, MeshCNN [20] utilized rotation- and translation-invariant geometric features of the mesh edges. In contrast, our method leverages rotation- and translation-invariant nodes and edge features, along with directional and positional information that varies with geometric transformation. We accomplish this through a novel graph neural network framework integrating mesh-specific geometric insights.

3. Method

This section describes RIMeshGNN and the message-passing graph neural network (GNN) [2, 16, 43] used to encode a 3D shape that remains invariant to $E(3)$ transformations. In our context, all shapes are considered to be represented by watertight, 2-manifold meshes.

3.1. E(3)-Equivariant GNN

A 3D shape is defined as a mesh comprised of vertices, edges, and faces. This mesh can be represented as a graph, denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_i\}_{i=1}^{N}$ is the set of N vertices in the mesh, $\mathcal{E} = \{e_{ij}\}$ is the set of E edges, where edge e_{ij} connects two vertices v_i and v_j. Each vertex v_i, edge e_{ij}, and the entire graph $\mathcal{G}$ are associated with feature vectors f_i, h_{ij}, and $h_{\mathcal{G}}$. Vertex features include rotation equivariant properties, such as vertex coordinates x_i and normal n_i, and rotation invariant feature h_i, such as mean and Gaussian curvature, area, and dihedral angle ($f_i = (x_i, n_i, h_i)$). We define the vertex area as one-third of the total face area of its 1-ring neighborhood. The vertex dihedral angle represents the maximum dihedral angle of all edges incident to the vertex. Edge features (h_{ij}) are rotation invariant, including dihedral and internal angles and edge-to-height ratios. The initial graph feature $h_{\mathcal{G}}$ includes a histogram of the lengths and dihedral angles of all the edges of the mesh. Figure 1 illustrates the input features for the vertex, edge and mesh graph.

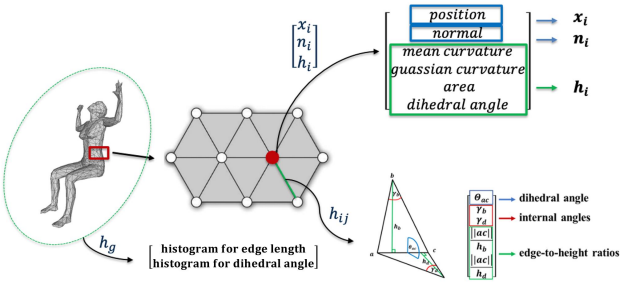

Figure 1. Input feature vector for the vertex, edge and mesh graph.

Our proposed E(3) equivariant layer utilizes the nodes,

edges, and graph features as input and generates corresponding output features by gathering information from the neighboring nodes in the mesh graph through nonlinear transformation and permutation-invariant aggregation functions. To achieve sufficient expressive power, a learnable transformation must be employed to convert the input features into higher-level features. Specifically, we apply a shared transformation $\Phi_e : \mathbb{R}^{F'} \rightarrow \mathbb{R}^F$, *i.e.* a feed-forward neural networks, to each edge, utilizing the edge's and its vertices' features, and the graph embedding as inputs:

$$h_{ij}^{l+1} = \Phi_e(h_{ij}^l, h_i^l, h_j^l, \|x_i^l - x_j^l\|_2, n_i^l \cdot {n_j^l}^T, h_{\mathcal{G}}^l), \quad (1)$$

where h_{ij}^l is the edge embedding and h_i^l and h_j^l represent the embeddings of the rotation invariant portion of the two incident vertices. $\|x_i^l - x_j^l\|_2$ is the distance between the incident vertices' coordinates. $n_i^l \cdot {n_j^l}^T$ is the cosine of angles between the incident vertices' normal and $h_{\mathcal{G}}^l$ denotes the graph embedding.

Furthermore, we split the node features update into rotation equivariant and invariant portions. Particularly, we update the vertex coordinates and normal using the weighted sum of the relative differences between the vertex and its neighboring nodes. The weights are separately computed using the functions Φ_x and Φ_n, and are approximated by Multilayer Perceptrons (MLPs), which take the edge embedding h_{ij} corresponding to the relative differences $(x_i - x_j)$ and $(n_i - n_j)$ and generate scalar values $\Phi_x(h_{ij})$ and $\Phi_n(h_{ij})$, as follows:

$$x_i^{l+1} = x_i^l + \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} (x_i^l - x_j^l) \Phi_x(h_{ij}^{l+1}) \quad (2)$$

$$n_i^{l+1} = n_i^l + \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} (n_i^l - n_j^l) \Phi_n(h_{ij}^{l+1}). \quad (3)$$

The embedded normal vector is then normalized:

$$n_i^{l+1} = \frac{n_i^{l+1}}{\|n_i^{l+1}\|_2}. \quad (4)$$

To update the rotation invariant component of the node feature h_i, we utilize a learnable transformation Φ_h approximated by an MLP that operates on each vertex v_i. This transformation takes as input the aggregation of edge updates for edges that are incident to vertex v_i, the initial feature h_i^l, and the entire graph feature $h_{\mathcal{G}}^l$, as follows:

$$h_i^{l+1} = \Phi_h(h_i^l, \sum_{j \in \mathcal{N}_i} h_{ij}^{l+1}, h_{\mathcal{G}}^l). \quad (5)$$

We have opted to use the summation function for aggregation, and our various experiments and ablation studies have provided supporting evidence for this choice.

The global graph attribute $h_{\mathcal{G}}$ is updated using a learnable transformation $\Phi_{\mathcal{G}}$. This transformation takes the aggregation of node updates $s_{\mathcal{G}}^{l+1}$, edge updates h_{ij}^{l+1}, and the graph attributes of the previous layer $h_{\mathcal{G}}^l$ as input. The result is an updated feature vector for $h_{\mathcal{G}}^{l+1}$, defined as:

$$h_{\mathcal{G}}^{l+1} = \Phi_g(s_{\mathcal{G}}^{l+1}, \frac{1}{E} \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} h_{ij}^{l+1}, h_{\mathcal{G}}^l). \quad (6)$$

In order to calculate the aggregation of node features $s_{\mathcal{G}}^{l+1}$, we have designed a novel aggregation function for combining node features, which will be discussed in the next section. Additionally, we have chosen the MEAN function for edge feature aggregation, which is supported by the results of our numerous experiments and ablation studies.

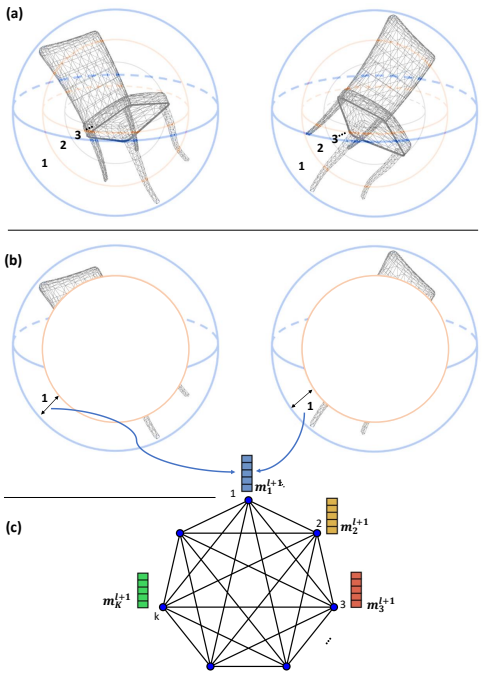

Figure 2. Permutation and Rotation Invariant Aggregation Function: (a) illustrates the spherical bins and bucketizing process. (b) demonstrates the rotation-invariance property of the process, and the mesh vertices associated with the first bin and its corresponding feature vector m_1^{l+1}, (c) presents the complete graph with n_{bin} nodes, where each node represents a spherical region.

3.2. Permutation, Rotation and Reflection Invariant Aggregation Function

After the input 3D shapes have been scaled to fit inside a unit sphere centered at the origin, we proceed to bucketize the nodes $\mathcal{V}$ into n_{bin} groups based on their distance from the origin. This process will assign an index k_i to each node v_i. To achieve this, we divide the range of distances from 0

to 1 into equal intervals, creating spherical bins illustrated in Figure 2a and expressed by the following equation:

$$k_i = \textit{Bucketize}(\|x_i\|, n_{bin}). \tag{7}$$

The node embeddings h_i^{l+1} that belong to each bin are averaged to form m_k^{l+1}, which represents the bin:

$$m_k^{l+1} = \frac{1}{|\{v_i | k_i = k\}|} \sum_{k_i=k} h_i^{l+1} \quad k \in \{1, 2, \ldots, n_{bin}\}. \tag{8}$$

Figure 2b illustrates the mesh vertices associated with the first bin and its corresponding feature vector m_1^{l+1}. We construct a complete graph with n_{bin} nodes, where each node is represented by m_k^{l+1}, and apply a two-layer graph attention network (GAT) on the complete graph, as illustrated in Figure 2c. The GAT [51] assigns varying importance to the neighborhood of each node (spherical bin) in each layer, which enhances performance for some downstream tasks. We calculate a global feature vector $s_{\mathcal{G}}^{l+1}$ for the entire complete graph by aggregating the node (bin) embedding vectors using the MEAN function:

$$s_{\mathcal{G}}^{l+1} = \frac{1}{n_{bin}} \sum_{k=1}^{n_{bin}} \Phi_{GAT}(m_k^{l+1}). \tag{9}$$

This entire process serves as a permutation, rotation, and reflection invariant aggregation function that consolidates the rotation invariant node features of the input mesh. The aggregated node features will then be used as one of the inputs for computing global mesh attributes utilized for the shape classification task.

3.3. Analysis of E(3) Equivariance and Invariance Property

In this section, we explore the equivariance properties of our proposed GNN layer and aggregation function with respect to rotation and translation. We observe that certain geometric features, such as node features h_i (mean and Gaussian curvature, area, and dihedral angles), edge features h_{ij} (dihedral and internal angles, and edge-to-height ratios), and shape feature $h_{\mathcal{G}}$ (a histogram of the length and dihedral angles of all edges of the mesh) remain invariant to rotation and translation. Furthermore, the relative distance between two coordinates $\|x_i^0 - x_j^0\|_2$ and cosine similarity between two normal vectors are also invariant to rotation and translation. This is because the distance between nodes and the relative angles between adjacent normal vectors do not change when objects are rotated or translated. As a result, we can ensure that edge features generated by Equation 1 remain invariant to E(3) transformation.

Equation 2, similar to the one presented in [43], is computed as a weighted sum of differences between coordinates x_i and x_j, which is then added to x_i. Any rotation applied to input coordinates will result in an equivalent rotation in $x_i - x_j$ and x_i. Therefore, the output x_i^{l+1} undergoes an equivalent rotation as the input when the coordinates are rotated. Likewise, translating the coordinates will alter x_i, but not alter the coordinate difference and thus cause an equivalent translation in the output. The same principle applies to Equation 3.

Moreover, indices k_i are assigned to the initial coordinates of shapes, which have already been translated and scaled to fit in a unit sphere centered at the origin, as illustrated in Equation 7. The process of bucketizing is rotation invariant as it depends on the nodes' distance to the center and remains unchanged when shapes are rotated. Equation 8 averages the invariant feature vectors of nodes within the same spherical shell. Since bucketization is rotation invariant, the averaging of invariant features remains unchanged under E(3) transformations. Equation 9 computes graph features based on the bin features m_k^{l+1}. Since m_k^{l+1} is invariant to E(3) transformations, Equation 9 is also invariant to E(3) transformations. Figure 2b illustrates this characteristic. Finally, Equation 6 maintains invariance due to its inputs' translation and rotation invariance properties. To summarize, the composition of our proposed GNN layer and aggregation functions exhibits E(3) equivariance inductively. For classification we utilize the graph attribute obtained through Equation 6, which is E(3) invariant.

3.4. Local Pooling Layer

We introduce a local pooling layer for RIMeshGNN that performs mesh coarsening at various scales; thus allowing for mesh analysis at multiple resolutions. Our method is influenced by Garland and Heckbert's surface simplification algorithm [15] and MeshCNN [20] and employs edge contraction as the basis for our pooling technique. We modify and extend Diehl's edge pooling method [11] for a geometric mesh context to develop a learnable pooling strategy.

The pooling layer performs a series of edge collapse operations, where each collapse operation results in an edge collapsing to a vertex, and subsequently, two pairs of edges from its adjacent faces merge into two edges, as demonstrated in Figure 3. To identify the most appropriate edges for edge collapses within each pooling layer, we use the updated edge embeddings generated by the prior GNN layer in Equation 1 (h_{ij}^{l+1}). A score is computed for each edge using a single fully connected layer with *softmax* as a non-linear activation function:

$$S_{ij} = \Phi_{score}(h_{ij}^{l+1}). \tag{10}$$

We first rank all edges according to their scores. The highest-scoring edge is chosen for the initial edge contraction. We proceed by choosing the next highest-scoring edge if its vertices have not been involved in prior contractions

Figure 3. Local pooling layer based on edge collapse.

within the current pooling layer; otherwise, we skip that edge. This precaution is taken to prevent redundancy and optimize the efficiency of the process. Specifically, by deferring the update of the mesh graph's edges until after all contractions are completed, the overall computational time is reduced, minimizing the need for repeated updates after each individual contraction.

We continue until this process reaches the end of the ranked list. This approach allows the network to collapse the least important parts of the mesh while preserving the more relevant regions in relation to the network's purpose enforced by the loss function. It is important to recognize that not all edges can be collapsed. Our approach prohibits edge collapses that result in non-manifold faces, e.g. those that produce faces with zero area. Also, an edge is not suitable for collapse if the intersection of the 1-ring vertices of the edge's end nodes contains three or more vertices.

More specifically, in each edge collapse operation, an edge $e_{ij} = (v_i, v_j)$ is contracted to a new vertex v_k. This action also removes two pairs of edges namely $(e_{mi} = (v_m, v_i), e_{mj} = (v_m, v_j)$ and $(e_{ni} = (v_n, v_i), e_{nj} = (v_n, v_j))$ and introduces $e_{mk} = (v_m, v_k)$ and $e_{nk} = (v_n, v_k)$. We illustrate this process in Figure 3. To determine the feature of $e_{mk} = (v_m, v_k)$, we take the average of the features of edges e_{mi} and e_{mj} ($h_{mk}^{l+1} = 0.5 \cdot (h_{mi}^{l+1} + h_{mj}^{l+1})$). Similarly, the feature of e_{nk} is computed by averaging the features of edges e_{ni} and e_{nj}. Additionally, we combine the features of the two collapsed nodes v_i and v_j to generate new vertex features for v_k:

$$h_k^{l+1} = S_{ij} \cdot (h_i^{l+1} + h_j^{l+1}) \tag{11}$$

$$x_k^{l+1} = 0.5 \cdot (x_i^{l+1} + x_j^{l+1}) \tag{12}$$

$$n_k^{l+1} = \frac{n_i^{l+1} + n_j^{l+1}}{\|(n_i^{l+1} + n_j^{l+1}\|_2} \tag{13}$$

$$x_k = 0.5 \cdot (x_i + x_j). \tag{14}$$

Following the approach in [11], to ensure that the gradient can flow into the scores, we multiply the combined node features by the edge score S_{ij} in Equation 11. Our proposed local pooling layer guarantees that the pooling process maintains a watertight and manifold shape throughout the process. Furthermore, it is rotation-equivariant and does not affect the rotation-invariance characteristic of our classification network.

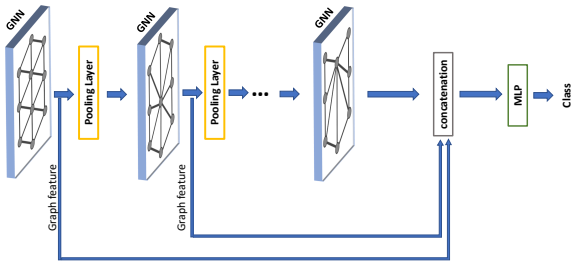

Figure 4. RIMeshGNN architecture.

3.5. Network Architecture

Our classification network employs multiple GNN and pooling layers to capture node and edge dependencies over long distances and to facilitate information propagation across multiple hops. Nevertheless, stacking several GNN layers can cause over-smoothing and performance issues in deep networks or graphs with small diameters [6, 58]. To address the problem of over-smoothing and enhance model capacity, we concatenate the graph features from all layers, thereby directly fusing all the information from intermediate layers. The embedding vector produced for nodes, edges, and graphs can serve various mesh analysis purposes, including mesh classification, shape retrieval, and segmentation. In order to accomplish graph classification, we incorporate an MLP sub-network as the final layer. The proposed neural network is then trained end-to-end to achieve mesh classification by minimizing cross-entropy loss over all predicted categories and labeled graphs. Figure 4 provides an overview of the proposed classification network.

4. Experiments and Results

Our proposed E(3)-equivariant Graph Neural Network and rotation-invariant aggregation layer have a broad range of applications in 3D shape analysis, due to their flexibility and adaptability. These components form the foundational elements of our E(3) invariant classification network. We quantitatively evaluated the network's performance for a shape classification task and compared the results with the current state-of-the-art outcomes. The RIMeshGNN model uses PyTorch [38] as the deep learning framework on a computer with a 3.8 GHz CPU (24 cores), a 64 GB RAM, and a TITAN RTX 24GB GPU. The graph neural network operations are performed with the Deep Graph Library (DGL) [53]. We have conducted extensive experiments to demonstrate the effectiveness of the key components of our proposed GNN layer and model.

Dataset	SHREC11	
Split	Split 16	Split 10
Evaluation Process	NR/NR	NR/NR
GWCNN [13]	96.6%	90.3%
MeshCNN [20]	98.6%	91.0%
MeshWalker [24]	98.6%	97.1%
HodgeNet [46]	99.1%	94.6%
SubdivNet [21]	99.9%	99.5%
RIMeshGNN (Ours)	**100%**	**100%**

Table 1. Experimental Results for SHREC11 classification.

4.1. Data Preprocessing and Augmentation

Within our framework, it is assumed that all shapes are represented as water-tight manifold meshes. However, many 3D shapes found in various datasets are not water-tight manifolds. As a result, the ManifoldPlus [22] method is used in a preprocessing stage to convert the triangular meshes into water-tight manifolds. While our method does not mandate meshes to have the same number of vertices or faces, we simplify all meshes to precisely 750 faces to reduce the computational cost of training. In addition, to reduce network sensitivity to size, we translate and scale all 3D meshes to ensure they fit within a unit sphere that is centered at the origin.

Augmentation. Two distinct methods for data augmentations are utilized: 1) adding varying Gaussian noise to each vertex of the shape, and 2) randomly displacing vertices to various locations on the surface of the mesh within a close-to-planar surface region [20]. As our classification network is rotation invariant, including rotated 3D shapes in the dataset does not help in network training.

4.2. Classification

We evaluated our proposed graph neural network (GNN) and aggregation function for mesh classification on two datasets, namely SHREC11 [30] and ModelNet40 [56], and conducted three separate experiments to demonstrate the effectiveness and superiority of our approach. Initially, the classification network was trained using the original datasets that contained aligned shapes (No random rotation, NR). The test process was divided into two groups. In the first experiment, the classification network predicted category labels for the original datasets with aligned test samples (NR). In the second experiment, category labels were predicted by the classification network after applying an arbitrary rotation transformation to the test data (AR). This experiment simulated real-world scenarios where the 3D mesh can be represented in any coordinate system, and the shape orientation is unknown to the network. In the third experiment, we examined the effect of augmentation on rotation-sensitive baseline methods. We expanded our investigation to include training the baseline methods using augmented data generated through random rotations. The subsequent evaluation aimed to assign labels to arbitrarily rotated test samples (AR/AR).

SHREC11. The dataset introduced by Lian *et al.* [30] contains 30 different classes, each with 20 samples. Following the setup in [13, 20, 34], we perform training and evaluation on two types of dataset splits: 1) Split 16: where the samples of each class are randomly split into 16 training examples and 4 testing examples, 2) Split 10: where the samples of each class are randomly split into 10 training examples and 10 testing examples.

Our method is compared to multiple state-of-the-art deep learning methods that utilize a mesh representation of 3D shapes as input. The evaluation results are shown in Table 1. Our classification network is rotation invariant and achieves an outstanding 100% accuracy in the NR/NR evaluation setting for both split 16 and split 10. It maintains this accuracy even when an arbitrary rotation is applied to the test samples in the NR/AR evaluation setting for both data splits. Our method outperforms all compared methods on aligned and unaligned data. The unavailability of the methods' code or pre-trained models for the SHREC dataset limited our ability to perform evaluations of the models within the NR/AR and AR/AR evaluation settings. The models GWCNN [13], MeshWalker [24], HodgeNet [46], and SubdivNet [21], which utilize xyz and normal feature vectors, are vulnerable to the impact of rotation. Consequently, their performance suffers when evaluated on datasets that include unknown rotation transformations in the NR/AR assessment framework. In contrast, MeshCNN [20] is capable of handling random rotation due to its use of solely rotation- and translation-invariant edge features. Our proposed approach takes advantage of both rotation-sensitive and rotation-invariant features of nodes and edges.

ModelNet40. The ModelNet40 dataset contains 12,311 shapes categorized into 40 different classes. The dataset was introduced by Wu *et al.* [56], with 9,843 models used for training and 2,488 models for testing. ModelNet40 has become a widely used benchmark for 3D geometric learning, particularly in tasks related to classification and retrieval. However, it should be noted that the majority of the shapes within the dataset are not watertight manifolds. To address this issue, we reconstructed the shapes in ModelNet40, producing manilfold meshes using ManifoldPlus [22] and then simplifying them to 750 faces using Garland's method [15]. We refer to this modified dataset as Manifold40. Due to reconstruction errors and simplification distortions, Manilfold40 meshes present a more challenging classification task, resulting in slightly lower accuracies compared to the results from ModelNet40 meshes.

We compare our model to several state-of-the-art deep learning techniques that employ triangular mesh representa-

Dataset	Manifold40** (ModelNet40)				# Parameters
Evaluation Setting	NR/NR †	NR/AR	AR/AR	Input	
MeshNet [14]	88.4% (91.9%)	10.8% (11.0%)	81.1%	Mesh	4.2 M
MeshWalker [24]	90.5% (**92.3%**)	11.2% (11.4%)	- -	Mesh	12.6 M
SubdivNet [21]	**91.2%**	10.2%	87.0%	Mesh	0.8 M
Laplacian2Mesh [12]	90.9%	-	-	Mesh	-
RIMeshGNN (Ours)	90.7%	**90.7%**	**90.7%**	Mesh	0.9 M
MeshWalker* [24]	(**94.4%**)	(13.4%)	- -	Mesh	12.6 M
Laplacian2Mesh* [12]	92.8%	-	-	Mesh	-
RIMeshGNN* (Ours*)	**93.4%**	**93.4%**	**93.4%**	Mesh	0.9 M
DynamicGCN [54]	93.5%	16.6%	81.1%	point cloud	-
DeltaConv [55]	**93.8%**	15.6%	- -	point cloud	-
ClusterNet [8]	87.1%	87.1%	87.1%	point cloud	-
Pose-Selector [28]	90.2%	**90.2%**	**90.2%**	point cloud	-
VN-DGCN [10]	90.0%	90.0%	90.0%	point cloud	-

Table 2. Experimental Results on Manifold40 (ModelNet40) Classification. The dash (-) indicates that the corresponding method's code is unavailable, preventing the evaluation of the model under the NR/AR and AR/AR settings and further examination of the network architecture. - - denotes our inability to train the data with rotation augmentation due to the substantial computational demands of training or data preprocessing. The accuracy of models that can handle non-watertight and non-manifold 3D shapes from the ModelNet40 dataset is denoted by values in parentheses. * The reported accuracy excludes five cross-labeled classes (desk/table and plant/flower-pot/vase). †The accuracy values for models utilizing mesh input on the Manifold40 dataset are reported from [12, 21] .

tions of 3D objects. Table 2 shows that our method achieves comparable results to prior work in the NR/NR evaluation scenario, where both the training and testing are performed on aligned dataset samples. Our approach, in particular, reaches a classification accuracy of 90.7%. In our analysis, we identified some categories with cross-labeling issues, such as desk/table and plant/flowerpot/vase, which contain a notable number of cross-labeled samples. Following the approach of MeshWalker [24] and Laplacian2Mesh [12], we excluded the models of these five categories. As a result, our classification accuracy reached 93.4%, a result that falls between the accuracy of Laplacian2Mesh and MeshWalker. In NR/AR setup, we maintain the classification accuracy of 90.7% on the randomly rotated full test set and significantly outperform all compared methods. The accuracy of RIMeshGNN also remains at 93.4% for the reduced dataset in the NR/AR setup. In contrast, the majority of other deep learning approaches that incorporate Euclidean coordinates and normal vectors struggle to generalize to unfamiliar orientations, resulting in a significant decline in performance when assessed in the NR/AR evaluation setting. It could be argued that rotation augmentation might enhance the performance of rotation-sensitive models [14,18,21,24,42,54,55]; however, this requires training with extensive data augmentation, substantially more complex analysis models, and increased computation resources, while our proposed method does not require any rotation augmentation. Furthermore, our method achieves comparable accuracy in the NR/NR evaluation setup, and also maintains consistent accuracy in the NR/AR and AR/AR configuration, where other models struggle, all while using significantly fewer learnable parameters.

Furthermore, we compare our approach with state-of-the-art deep learning methods that utilize point cloud representation for 3D objects. While rotation-sensitive methods [42, 54, 55] experience a substantial accuracy drop in NR/AR setting, Clusternet [8], VN-DGCN [10] and PoseSelector [28] achieve rotation-invariant classification. Notably, the accuracy of PoseSelector [28] decreases in classification tasks involving datasets with non-symmetrical objects, where intra-class shapes do not share similar structures. However, our proposed method consistently achieves state-of-the-art performance in accuracy.

5. Conclusions

In this paper, we present a novel representation learning model, RIMeshGNN, to address the challenge of rotation invariance in shape analysis tasks. Our approach consists of a E(3) equivariant GNN layer, aggregation function, and local pooling layer, which together form the basis of our proposed rotation-invariant network for mesh model classification. The method enables the analysis of mesh-structured data while maintaining accuracy even when test samples experience arbitrary rotations, without requiring extensive training on rotation-augmented dataset. Comprehensive tests on various datasets demonstrate that our approach achieves state-of-the-art performance. Additionally, the proposed representation learning framework, can and will be adapted for other mesh analysis tasks, such as mesh segmentation and retrieval.

References

[1] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gauzere, Sebastien Adam, and Paul Honeine. Bridging the gap between spectral and spatial domains in graph neural networks. *arXiv preprint arXiv:2003.11702*, 2020. 3

[2] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks, 2018. 3

[3] Ran Ben Izhak, Alon Lahav, and Ayellet Tal. AttWalk: Attentive cross-walks for deep mesh analysis. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1546–1555, 2022. 1, 2

[4] Giorgos Bouritsas, Sergiy Bokhnyak, Stylianos Ploumpis, Michael Bronstein, and Stefanos Zafeiriou. Neural 3D morphable models: Spiral convolutional networks for 3D shape representation learning and generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7213–7222, 2019. 2

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 3

[6] Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv preprint arXiv:2006.13318*, 2020. 6

[7] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 1

[8] 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/CVF conference on computer vision and pattern recognition*, pages 4994–5002, 2019. 8

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016. 3

[10] Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209, 2021. 3, 8

[11] Frederik Diehl, Thomas Brunner, Michael Truong Le, and Alois Knoll. Towards graph pooling by edge contraction. In *ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data*, 2019. 5, 6

[12] Qiujie Dong, Zixiong Wang, Manyi Li, Junjie Gao, Shuangmin Chen, Zhenyu Shu, Shiqing Xin, Changhe Tu, and Wenping Wang. Laplacian2Mesh: Laplacian-based mesh understanding. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 8

[13] Danielle Ezuz, Justin Solomon, Vladimir G Kim, and Mirela Ben-Chen. GWCNN: A metric alignment layer for deep shape analysis. In *Computer Graphics Forum*, volume 36, pages 49–57. Wiley Online Library, 2017. 7

[14] Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao. MeshNet: Mesh neural network for 3D shape representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8279–8286, 2019. 3, 8

[15] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997. 5, 7

[16] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017. 3

[17] Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. SpiralNet++: A fast and highly efficient mesh convolution operator. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[18] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021. 1, 2, 8

[19] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017. 3

[20] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. MeshCNN: A network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 3, 5, 7

[21] Shi-Min Hu, Zheng-Ning Liu, Meng-Hao Guo, Jun-Xiong Cai, Jiahui Huang, Tai-Jiang Mu, and Ralph R Martin. Subdivision-based mesh convolution networks. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022. 1, 3, 7, 8

[22] Jingwei Huang, Yichao Zhou, and Leonidas Guibas. ManifoldPlus: A robust and scalable watertight manifold surface generation method for triangle soups. *arXiv preprint arXiv:2005.11621*, 2020. 7

[23] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. RotationNet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[24] Alon Lahav and Ayellet Tal. MeshWalker: Deep mesh understanding by random walks. *ACM Transactions on Graphics (TOG)*, 39(6):1–13, 2020. 1, 2, 7, 8

[25] Itai Lang, Asaf Manor, and Shai Avidan. SampleNet: Differentiable point cloud sampling. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7578–7588, 2020. 1

[26] Truc Le and Ye Duan. PointGrid: A deep network for 3D shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018. 1

[27] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. CayleyNets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018. 3

[28] Feiran Li, Kent Fujiwara, Fumio Okura, and Yasuyuki Matsushita. A closer look at rotation-invariant deep point cloud analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16218–16227, 2021. 3, 8

[29] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018. 1

[30] Z Lian, A Godil, B Bustos, M Daoudi, J Hermans, S Kawamura, Y Kurita, G Lavoua, P Dp Suetens, et al. Shape retrieval on non-rigid 3D watertight meshes. In *Eurographics workshop on 3D object retrieval (3DOR)*. Citeseer, 2011. 7

[31] Isaak Lim, Alexander Dielen, Marcel Campen, and Leif Kobbelt. A simple approach to intrinsic correspondence learning on unstructured 3D meshes. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 2

[32] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on Riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshop*, pages 37–45, 2015. 2

[33] Daniel Maturana and Sebastian 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. 1, 2

[34] Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. Primal-dual mesh convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 952–963. Curran Associates, Inc. 1, 3, 7

[35] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017. 1, 2, 3

[36] Federico Monti, Oleksandr Shchur, Aleksandar Bojchevski, Or Litany, Stephan Günnemann, and Michael M Bronstein. Dual-primal graph convolutional networks. *arXiv preprint arXiv:1806.00770*, 2018. 3

[37] A. A. M. Muzahid, Wanggen Wan, Ferdous Sohel, Naimat Ullah Khan, Ofelia Delfina Cervantes Villagómez, and Hidayat Ullah. 3D object classification using a volumetric deep neural network: An efficient octree guided auxiliary learning approach. *IEEE Access*, 8:23802–23816, 2020. 1, 2

[38] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 6

[39] Adrien Poulenard and Leonidas J Guibas. A functional approach to rotation equivariant non-linearities for tensor field networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13174–13183, 2021. 3

[40] 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*, pages 652–660, 2017. 1, 2

[41] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas 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. 1, 2

[42] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 1, 2, 8

[43] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *International Conference on Machine Learning*, pages 9323–9332. PMLR, 2021. 3, 5

[44] Bahareh Shakibajahromi, Saeed Shayestehmanesh, Daniel Schwartz, and Ali Shokoufandeh. HyNet: 3D segmentation using hybrid graph networks. In *2021 International Conference on 3D Vision (3DV)*, pages 805–814. IEEE, 2021. 1, 3

[45] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022. 2

[46] Dmitriy Smirnov and Justin Solomon. HodgeNet: Learning spectral geometry on triangle meshes. *ACM Transactions on Graphics (TOG)*, 40(4):1–11, 2021. 7

[47] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. 2

[48] 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*, pages 980–988, 2019. 2, 3

[49] Zhiyong Tao, Yixin Zhu, Tong Wei, and Sen Lin. Multi-head attentional point cloud classification and segmentation using strictly rotation-invariant representations. *IEEE Access*, 9:71133–71144, 2021. 2, 3

[50] 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. 2

[51] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 3, 5

[52] Nitika Verma, Edmond Boyer, and Jakob Verbeek. FeaStNet: Feature-steered graph convolutions for 3D shape analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2598–2606, 2018. 1, 3

[53] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019. 2, 6

[54] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph CNN for learning on point clouds.(2018). *arXiv preprint arXiv:1801.07829*, 222, 2018. 1, 2, 8

[55] Ruben Wiersma, Ahmad Nasikun, Elmar Eisemann, and Klaus Hildebrandt. Deltaconv: anisotropic operators for geometric deep learning on point clouds. *ACM Transactions on Graphics (TOG)*, 41(4):1–10, 2022. 1, 2, 8

[56] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 1, 7

[57] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional ShapeContextNet for point cloud recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4606–4615, 2018. 1

[58] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR, 2018. 6

[59] Y. You, Y. Lou, R. Shi, Q. Liu, Y. Tai, L. Ma, W. Wang, and C. Lu. PRIN/SPRIN: On extracting point-wise rotation invariant features. *IEEE Transactions on Pattern Analysis amp; Machine Intelligence*, 44(12):9489–9502, 2022. 1

[60] 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)*, pages 204–213. IEEE, 2019. 2

[61] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. RI-Conv++: Effective rotation invariant convolutions for 3D point clouds deep learning. *International Journal of Computer Vision*, 130(5):1228–1243, 2022. 3