L^2-GNN: Graph Neural Networks with Fast Spectral Filters Using Twice Linear Parameterization

Siying Huang[a], Xin Yang[a], Zhengda Lu[b,*], Hongxing Qin[a], Huaiwen Zhang[c], Yiqun Wang[a,*]

[a]*Chongqing University, Chongqing 400044, China*
[b]*University of Chinese Academy of Science, Beijing 100049, China*
[c]*Inner Mongolia University, Hohhot 010021, China*

Abstract

To improve learning on irregular 3D shapes, such as meshes with varying discretizations and point clouds with different samplings, we propose L^2-GNN, a new graph neural network that approximates the spectral filters using twice linear parameterization. First, we parameterize the spectral filters using wavelet filter basis functions. The parameterization allows for an enlarged receptive field of graph convolutions, which can simultaneously capture low-frequency and high-frequency information. Second, we parameterize the wavelet filter basis functions using Chebyshev polynomial basis functions. This parameterization reduces the computational complexity of graph convolutions while maintaining robustness to the change of mesh discretization and point cloud sampling. Our L^2-GNN based on the fast spectral filter can be used for shape correspondence, classification, and segmentation tasks on non-regular mesh or point cloud data. Experimental results show that our method outperforms the current state of the art in terms of both quality and efficiency.

***Keywords:** Graph Neural Networks, Spectral Filters, Linear Parameterization, Irregular 3D Shapes*

1. Introduction

In recent years, research on three-dimensional (3D) shape data such as meshes and point clouds has received increasing attention due to their richer information to represent the real world. With the success of convolutional neural networks (CNNs) in analyzing images and videos, many researchers have explored how to apply CNNs to 3D shapes. However, 3D shape data is irregular, and no common method currently exists for applying convolution to 3D shapes.

A good CNN on 3D shapes should satisfy the following three criteria: (1) The convolution operator should have a strong ability to aggregate shape information and obtain more meaningful information from shape data. (2) The convolution operator should be robust to the irregularity of different shapes, such as different discretizations of meshes and different samplings of point clouds. (3) The convolution filter should have low computational complexity to achieve good scalability for large-scale 3D data.

Some mesh-based methods [1, 2] mimic the idea of CNNs on two-dimensional (2D) data and apply local parameterization spatially on the mesh. Although these methods have achieved higher performance in shape correspondence tasks, they cannot be applied to point clouds due to the need for local parameterization. MeshCNN [3] proposes a convolution operation defined on mesh edges for shape classification and segmentation while it is sensitive to the change of mesh discretization as the meshes with different discretizations have different edge connectivity. Some recent approaches [4–6] require computing intrinsic information to handle different discretizations on meshes, leading to an enormous time cost. Point-based methods [7–10] often have strong information aggregation capabilities due to the carefully designed convolution operators, but these methods usually do not consider the effects of point cloud sampling.

Graph neural networks are another type of approach for handling 3D shapes. SpectralCNN [11] proposes to use spectral filters for convolution in the graph frequency domain, but this approach performs poorly due to the loss of locality of the convolution kernel, and computing spectral transform requires high computational complexity. ChebyGCN [12] utilizes the Chebyshev polynomials to parameterize spectral filters and achieves faster convolution speed. SplineCNN [13] uses B-splines to directly weight neighborhoods in the graph spatial domain. However, these acceleration methods cannot satisfy the robustness of networks to mesh discretization because the acquisition of local information is highly related to mesh discretization. Some works [14–16] introduce graph wavelets [17] into graph neural networks. However, these methods either cannot be applied to scenarios with different discretizations [14] or require computing wavelet functions [15, 16], which

*Corresponding authors

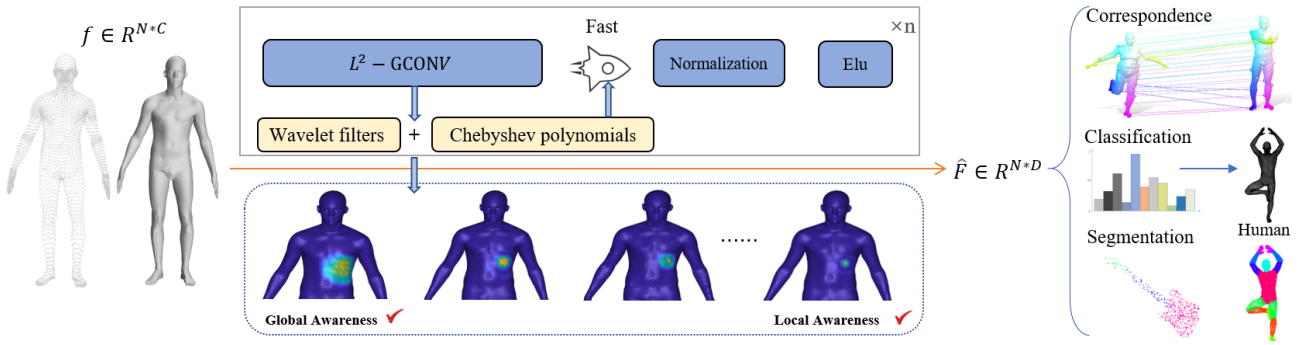


Figure 1. The overview of our L^2-GNN network which inputs feature $f \in R^{N\times C}$ of 3D shapes and outputs feature $\widehat{F} \in R^{N\times D}$. Employing an overall twice-linear parameterization structure, in the L^2-GCONV layer, we utilize neural network parameters as coefficients for wavelet basis functions to select wavelet filters with different scales for capturing low-frequency and high-frequency features. Additionally, we leverage Chebyshev polynomials to parameterize the basis functions of the wavelet filters, aiming to accelerate the training speed. Finally, we apply the output features to various tasks on data with different discretizations, achieving promising results.

require a very high computational complexity.

In this paper, we propose L^2-GNN, a fast graph neural network suitable for shape correspondence, classification, and segmentation tasks on both meshes and point cloud data. As shown in the Fig. 1, our idea is to use two linear parameterizations to approximate the spectral filter, while satisfying performance, robustness, and speed requirements at the same time. The basis functions of the first linear parameterization are the wavelet filter basis functions. By parameterizing the spectral filter with wavelet filter basis functions, we can not only maintain the local property of the filter but also incorporate global-aware information to further enhance the information aggregation ability of the convolutional operator. The basis functions of the second linear parameterization are the Chebyshev polynomial basis functions. We can speed up the convolution operation by parameterizing the wavelet filters with Chebyshev polynomials while maintaining the robustness to mesh discretization or point cloud sampling. Our experimental evaluations demonstrate that our graph neural network outperforms recent state-of-the-art methods on most tasks for both point clouds and meshes.

The main contributions of this work are summarized as follows:

- We present a new spectral filter based on two linear parameterizations that can enhance the information aggregation ability by increasing the receptive field while maintaining locality.
- Our proposed convolutional approach can reduce the computation complexity of convolution while maintaining robustness to the change of mesh discretization and point cloud sampling.
- L^2-GNN offers a new approach to address challenges in 3D deep learning, due to its ability to handle meshes and point clouds simultaneously and its strong performance across various tasks, such as shape correspondence, classification, and segmentation.

2. Related Work

There are many deep learning technologies applied to 3D shapes. We mainly review the work of 3D deep Learning and graph neural networks.

2.1. 3D Deep Learning

Traditional methods [18–23] typically use descriptors to represent features of 3D shapes, but the effectiveness of such methods needs improvement. The emergence of deep learning has made 3D deep learning possible. 3D deep learning methods are generally divided into two categories based on mesh and point cloud representations. Mesh-based methods are to apply learning methods on mesh representation. Some methods [1, 2, 13, 24–27] define convolution operations directly on manifolds. However, these methods need to consider the rotation of convolution operators in the tangent plane. Some methods [28–32] introduce the equivariant property in the convolution for the networks to address this problem. Subsequently, many works [5, 33–37] focus on extending the expressive power of the convolution operation. They propose to manipulate vector-valued data in local tangent space. [38,39] propose a multiscale framelet transform convolution method, which effectively reduces noise in nodes and structures, enhancing the robustness and flexibility of spectral graph neural networks in handling incomplete or perturbed graph data. MeshCNN [3] is a typical deep-learning framework specifically designed for processing polygonal meshes. It employs a novel mesh convolution operation on the local geometries of each edge and achieves good experimental results. Laplacian2Mesh [40] proposes to map the input mesh to the multi-dimensional Laplacian-Beltrami space, which can be used for the classification and

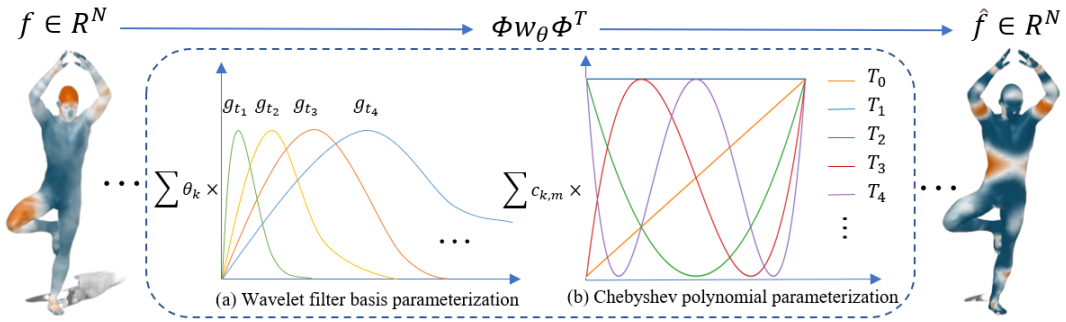


Figure 2. The illustration of our fast spectral filter leveraging twice-linear parameterization is as follows: With a single dimension features f as input, (a) we first adopt wavelet filter basis parameterization for the spectral filter, (b) then each wavelet filter basis is parameterized by the Chebyshev polynomials. Finally, we obtain the features $\hat{f}$ after convolution.

segmentation tasks.

On the other hand, point-based approaches typically focus on processing point cloud data. PointNet [7] is the first successful point cloud neural network, which directly learns features from point clouds. PointNet++ [8] then proposes a multi-scale architecture to further enhance the network on the tasks based on point clouds. DGCNN [9] dynamically updates the graph structure between layers to better learn semantic information of point sets. This method can extract local features of point clouds while maintaining vertex permutation invariance. Many recent works [10, 41–49] extend these traditional works by incorporating connectivity information, dynamic filters, cycle consistency and equivalent rigid transformations to further improve the performance. Recently, some methods [5, 6, 50, 51] can produce good learning effects on both mesh and point cloud. However, these methods are difficult to perform well on both mesh and point cloud data while maintaining efficiency.

2.2. Graph Neural Networks

Recently, graph neural networks have progressed rapidly, showing strong learning capabilities on irregular graphs and different applications [52–54]. As both meshes and point clouds can be easily converted into graph structures, graph neural networks have also been widely used for processing 3D shapes. Spectral CNN [11] first performs convolution operations on the graph spectral domain through the Fourier transform. ChebyGCN [12] approximate the spectral filter by Chebyshev polynomials with k order, which avoids spectral decomposition and speeds up the convolution process. GCN [55] further simplifies the spectral filter to 1 order and proposes a method for semi-supervised learning on graphs. Furthermore, GAT [56] combines a self-attention mechanism to aggregate neighboring nodes, achieving adaptive learning of different neighbor weights. SplineCNN [13] uses B-Spline kernel functions to perform convolution. The network achieves good performance, but due to the design of pseudo-coordinates on the edges, it does not have the invariant property of rigid transformation. LGCL [57] proposes a learnable graph convolution layer to perform regular convolution operations on graphs. ACSCNN [58] uses anisotropic Chebyshev polynomial kernels to achieve good performance on shape correspondence tasks. AMGCN [16] proposes to learn features trained by U-Net [59] to perform dense shape correspondence in an anisotropic (direction-sensitive) and multi-scale manner.

In addition to the Fourier spectral domain, wavelets have been used in various computer vision and machine learning applications, which can also be applied to graph neural networks, known as graph wavelets [17]. GWNN [14] uses the graph wavelet transform instead of the graph Fourier transform to extract features from graph signals. However, this method can only be used for graph structures with fixed discretization. To this end, MGCN [15] focuses on mesh structure and proposes a multi-scale graph convolutional network to learn compact and informative descriptors, which can maintain robustness to the change of mesh resolution while improving the discriminative power of descriptors. However, the method has low computational efficiency due to the complex computation of eigendecomposition for constructing the graph wavelets, and its performance has not been validated on point cloud data.

3. Method

3.1. Overview

Given a 3D shape represented by a mesh $M = (V, E)$ or a point cloud P, where $V = \{v_i | i = 1, ..., N\}$ and $P = \{p_i | i = 1, ..., M\}$ are 3D coordinates and $E = \{e_{i,j} | i, j = 1, ..., N\}$ is the set of edges on the mesh, our goal is to compute a feature $\mathrm{f}(v_i) \in \mathbb{R}^D$ for any given vertex v_i by our proposed graph neural network. Subsequently, the learned features can be processed using different frameworks to achieve correspondence, segmentation, and classification tasks. In the following sections, we first introduce the background of the graph neural network using one linear parametrization in Sec. 3.2. Then we describe our

proposed spectral filters using twice linear parametrization in Sec. 3.3. Finally, we discussed the details of our graph convolution layer and network architecture in Sec. 3.4.

3.2. Background

Graph neural networks are a type of neural network designed to work with non-Euclidean data that can output the node features. When the input feature is $f \in R^N$, a new feature $\hat{f} \in R^N$ can be calculated through graph convolution operation $T*f$. The convolution in the time domain is equal to the product in the frequency domain. The spectral convolution on the graph can be defined as follows.

$$\hat{f} = T*f = \Phi\left(\left(\Phi^T T\right) \odot \left(\Phi^T f\right)\right) = \Phi w_\theta \, \Phi^T f \qquad (1)$$

where w_θ represents the filter in the spectral domain, Φ is the matrix of eigenvector. This process involves projecting the graph signal f into the spectral domain as $\Phi^T f$, applying the filter w_θ to its frequency components, and then transforming the filtered signal back into the spatial domain by multiplying it with Φ, producing the final output $\hat{f}$.

One fundamental work is ChebyGCN [12], which uses one type of linear parametrization, the Chebyshev polynomial parametrization, to fit spectral filters. Since the spectral filter is represented by linear parametrization, the computational complexity can be significantly reduced, which has become the basis for many subsequent graph neural networks. In ChebyGCN, the spectral filter is expressed as follows.

$$w_\theta = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}) \qquad (2)$$

where T_k is a k-order Chebyshev polynomial and can be calculated using a recursive formula $T_k(x) = 2 * x * T_{k-1}(x) - T_{k-2}(x)$, and $\tilde{\Lambda} = \frac{2\Lambda}{\lambda_{\max}} - I_N$, which is a scaling matrix that scales the eigenvalue matrix Λ to a range of [-1,1] for using Chebyshev polynomial, where $\lambda_{\max}$ is the maximum eigenvalue of Λ.

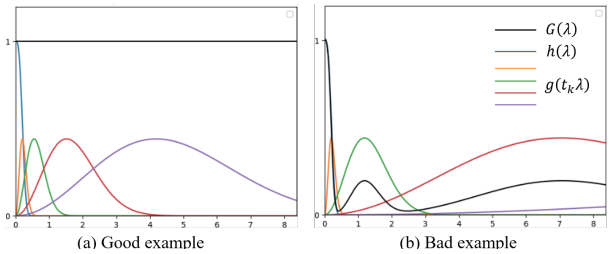
Due to linear parametrization, the eigenvalue matrix in the spectral filter w_θ can be combined with the eigenvector matrix to obtain the Laplacian matrix L as in Eq. 1, which avoids the computation of multiplying the eigenvalue matrix and largely reduces the complexity. The output features can be calculated as follows.

$$\hat{f} = \Phi w_\theta \Phi^T f \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}) f \qquad (3)$$

where $\tilde{L} = \Phi \tilde{\Lambda} \Phi^T = \frac{2L}{\lambda_{\max}} - I_N$.

3.3. Fast Spectral Filter Using Twice Linear Parameterization

ChebyGCN [12] only uses a single linear parameterization to obtain fast local spectral filters, but this also brings two issues. On the one hand, the receptive field of this convolutional method is local, and obtaining higher performance usually requires a larger receptive field. On the other hand, the spatial receptive field of this k-order polynomial filter is the k-ring neighborhood around the vertex, which makes it vulnerable to the effects of mesh discretization or point cloud sampling.



(a) Good example (b) Bad example

Figure 3. Two examples of classic wavelet filter basis functions: (a) Good example, (b) Bad example. We present the sum of squares $G(\lambda)$ (black curve), $h(\lambda)$ (blue curve) and $g(t_k\lambda)$ (other coloured curves). When $G \equiv 1$, the filter can better reconstruct the given signal.

To this end, our idea is to use twice linear parametrization as shown in Fig. 2. Our spectral filter is first linearly parameterized by the non-local wavelet filter basis function, and then the wavelet filter basis function is further linearly parameterized by the Chebyshev polynomial basis function. Since the wavelet function on shapes is independent of discretization and sampling, the proposed spectral filter is robust to mesh discretization or point cloud sampling. We further utilize Chebyshev polynomial to parameterize the wavelet filter basis functions, which reduce the computational complexity of the wavelet. In this way, the obtained spectral filter maintains the properties of fast processing, large spatial receptive fields, and discretization robustness at the same time. Next, we will first introduce the method of linear parameterization of spectral filters using wavelet filter basis functions, and then introduce the method of linear parameterization of wavelet filter basis functions using Chebyshev polynomial basis functions.

Wavelet filter basis functions are a set of orthogonal bases in the spectral domain. They are typically composed of a scaling filter function $h(\lambda)$, which captures the low-frequency components, and a set of wavelet filter functions $\{g(t_k\lambda)\}_{k=1}^K$ at different scales t_k, which represent the high-frequency components. Mexican hat functions are classic wavelet filter basis functions, as shown in Fig. 3. A good wavelet filter basis should satisfy the following constraint.

$$G(\lambda) = h^2(\lambda) + \sum_{k=1}^{K} g^2(t_k\lambda) \equiv 1 \qquad (4)$$

For simplification, we define $g_{t_k}(\lambda_j)$ to represent both

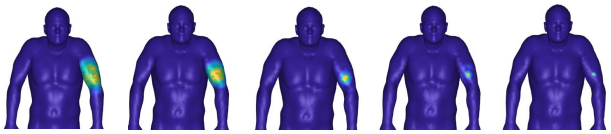


Figure 4. Our spectral filter is parameterized by the wavelet filter basis. We choose graph wavelet functions using five wavelet filters with different scales on one vertex to represent the spatial convolution. Compared with Chebyshev polynomial parameterization in ChebyGCN [12], our proposed method can capture low-frequency and high-frequency information simultaneously.

filter functions.

$$g_{t_k}(\lambda) = \begin{cases} h(\lambda), & if \quad k = 0 \\ g(t_k\lambda), & if \quad k > 0 \end{cases} \tag{5}$$

The spectral filter w_θ can be expressed on the wavelet filter basis functions as follows.

$$w_\theta = \sum_{k=0}^{K} \theta_k diag\left(\{g_{t_k}(\lambda_i)\}_{i=0}^{K}\right) = \sum_{k=0}^{K} \theta_k g_{t_k}(\Lambda) \tag{6}$$

where λ_i is the i^{th} eigenvalue of matrix Λ.

Given eigenvector matrix Φ, eigenvalue matrix Λ, and the wavelet filter basis functions $\{g_{t_k}(\lambda)\}_{k=0}^{K}$, the graph wavelet function $\psi_{t_k,v}$ with scale t_k on vertex v can be defined as follows [17].

$$\psi_{t_k,v} = \sum_{i=0}^{N-1} g_{t_k}(\lambda_i)\,\phi_i(v)\,\phi_i \tag{7}$$

where ϕ_i is the i^{th} eigenvector of matrix Φ, and $\phi_i(v)$ is the v^{th} component of ϕ_i, representing the response of vertex v at frequency λ_i.

The spectral convolution on the graph can then be derived as follows.

$$\hat{f} = T * f \approx \sum_{k=0}^{K} \theta_k \Psi_{t_k} f \tag{8}$$

where Ψ_{t_k} is a wavelet matrix composed of graph wavelet functions ψ_{t_k} with all scales $t_k = 0...K$.

Due to the multiscale nature of wavelet filter basis functions, our proposed convolution can capture low-frequency and high-frequency information on shape as shown in Fig. 4. However, to calculate $\Phi g_{t_k}(\Lambda)\,\Phi^T$, we need to apply eigendecomposition for the Laplacian matrix and obtain the eigenvector matrix and eigenvalue matrix, which is very time-consuming. To this end, we propose to use a second linear parameterization to approximate the wavelet filter basis functions g_{t_k} to achieve a fast spectral filter.

We choose Chebyshev polynomial basis for the second linear parameterization. The Chebyshev polynomials typically satisfy the following conditions. Given $T_0 = 1$, T_1=z, $z \in [-1, 1]$, $T_m(z)$ can be generated by the recursive relations: $T_m(z) = 2zT_{m-1}(z) - T_{m-2}(z)$. Any function $a(z)$ can be parameterized by the Chebyshev polynomial basis in an M-truncated form as follows.

$$a(z) = \frac{1}{2}c_0 + \sum_{m=1}^{M-1} c_m T_m(z) \tag{9}$$

The coefficients can be calculated in the Hilbert space of Chebyshev polynomial basis:

$$c_m = \frac{2}{\pi}\int_0^\pi \cos(m\theta)g(\cos(\theta))d\theta. \tag{10}$$

For our case, we have the Laplacian matrix L, the maximum eigenvalue $\lambda_{\max}$, and $\lambda \in [0, \lambda_{max}]$. We define $\overline{\lambda} = \frac{\lambda_{max}}{2}$ and z can be expressed as: $z = \frac{\lambda - \overline{\lambda}}{\overline{\lambda}} \in [-1, 1]$. In this case, our wavelet filter basis $g_{t_n}(\lambda)$ can be parameterized as follows.

$$g_{t_k}(\lambda) = g_{t_k}(\overline{\lambda}(z+1)) = \frac{1}{2}c_{m,0} + \sum_{m=1}^{M-1} c_{k,m} T_m\left(\frac{\lambda - \overline{\lambda}}{\overline{\lambda}}\right) \tag{11}$$

where $c_{k,m}$ is the coefficients for reconstructing our wavelet filter basis.

$$c_{k,m} = \frac{2}{\pi}\int_0^\pi \cos(m\theta)g_{t_k}((\overline{\lambda}(\cos(\theta)+1)))d\theta \tag{12}$$

In this way, given the form of the wavelet basis function, i.e., the Mexican hat function, we further utilize the Chebyshev polynomials to parameterize the wavelet filter basis functions.

We denote a shifted Chebyshev polynomials $\overline{T}_m(\lambda) = T_m(\frac{\lambda - \overline{\lambda}}{\overline{\lambda}})$, then the shifted recurrence relation is changed as follows.

$$\overline{T}_m(\lambda) = \frac{2}{\overline{\lambda}}(\lambda - 1)\left(\overline{T}_{m-1}(\lambda)\right) - \overline{T}_{m-2}(\lambda) \tag{13}$$

We first substitute the second linear parameterization (Eq. 11) into the first linear parameterization (Eq. 6):

$$w_\theta = \sum_{k=0}^{K} \theta_k g_{t_k}(\Lambda) = \sum_{k=0}^{K} \theta_k \left(\frac{1}{2}c_{m,0} + \sum_{m=1}^{M-1} c_{k,m}\overline{T}_m(\Lambda)\right), \tag{14}$$

then we substitute Eq. 14 into the spectral convolution formula (Eq. 1) and obtain our spectral convolution equation as follows.

$$\hat{f} = \Phi w_\theta\,\Phi^T f = \sum_{k=0}^{K} \theta_k \left(\frac{1}{2}c_{m,0} + \sum_{m=1}^{M-1} c_{k,m}\overline{T}_m(L)\right) f \tag{15}$$

where L can be computed using the cotangent Laplacian matrix on mesh or intrinsic Laplacian on point clouds [60].

For fast computing, we combine the shifted Chebyshev polynomials and the input feature in the recursive computation as follows.

$$\overline{T}_m(L)\,f = \frac{2}{\overline{\lambda}}(L - I)\left(\overline{T}_{m-1}(L)\,f\right) - \overline{T}_{m-2}(L)\,f \tag{16}$$

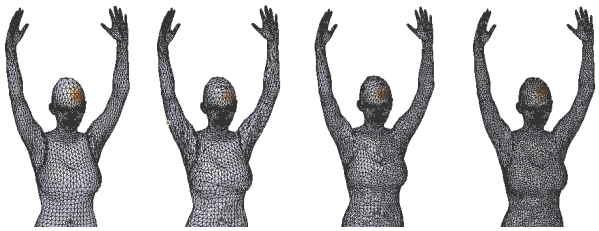


Figure 5. The illustration of the robustness of discretization. From left to right are four different cases of discretization. With the change of discretization, the receptive fields of the wavelet functions parameterized by the Chebyshev polynomial basis are similar, which indicates that our method can maintain robustness to discretization while improving the inference speed of graph convolution.

The use of twice linear parameterization in spectral filters enables acceleration of computation while maintaining robustness to discretization or sampling. Furthermore, the performance gains of L^2-GNN stem from the principled synergy of spectral localization and multi-scale feature learning. Specifically, our architecture incorporates Chebyshev polynomial-based wavelet approximation from ChebyGCN [12], while integrating the multi-scale convolution paradigm from MGCN [15]. As shown in Fig. 5 our filters achieve the same receptive field in four different discretizations. This is a property that ChebyGCN's spectral filters with one linear parameterization lack.

3.4. Graph Convolution Layer and Network Architecture

To avoid unnecessary calculation, we uniformly sample wavelet scales for each graph convolutional layer. Considering high-dimensional input feature $F \in R^{N\times C}$ and feature dimension transformations, we propose the following L^2-GCONV layer as follows.

$$\hat{F} = Norm\left(ELU\left(\sum_{k\in S_k}\left(\frac{1}{2}c_{m,0} + \sum_{m=1}^{M-1} c_{k,m}\overline{T}_m(L)\right) F\,\Theta_k\right)\right) \tag{17}$$

where $\Theta_k \in R^{C\times D}$ is the parameter to be learned, $\hat{F} \in R^{N\times D}$ is the output feature, and M is set to 50.

We use the proposed convolutional layer L^2-GCONV to construct our L^2-GNN. In addition to the first layer, we use five L^2-GCONV layers with 128-dimensional input and output features, followed by a multilayer perceptron with 96 output dimensions. We use an activation function ELU for each layer, which has been normalized by the batch norm layer. In addition, we choose 16 wavelet scales in our convolutional layer, which is discussed in Sec. 4.4. We use the cross-entropy loss as our training loss function for correspondence, classification, and segmentation tasks.

3.5. Discussion

SpectralCNN employs convolution operations in the graph frequency domain, however, it compromises the preservation of local features and incurs significant computational complexity. While ChebyGCN achieves faster convolution by parameterizing spectral filters with Chebyshev polynomials, it lacks robustness across different mesh discretizations. SplineCNN also maintains impressive training efficiency but its robustness diminishes when subjected to changes in resolution. ACSCNN applies anisotropic Chebyshev polynomial kernels, delivering strong performance in shape correspondence tasks. FC presents a surface convolution operator that enables robust and descriptive convolutions on surfaces. Nevertheless, neither of them maintains robustness when applied to varying resolutions. The central issue with graph convolution methods lies in developing filter representation that remain robust across different discretizations. We also focus on achieving high accuracy and maintaining low computational complexity. MGCN leads to high computational demands and low speed, although it maintains robustness when dealing with different resolutions. DiffusionNet is fast and robust to variant resolutions, but its accuracy needs to be improved. At the same time, multimodal-DFM boosts accuracy and maintains robustness to resolution changes. However, multimodal information negatively impacts processing speed. In conclusion, a reliable network that effectively balances

Table 1. Quantitative comparison results of shape correspondence on FAUST dataset. We show the average geodesic error ×100, which are computed on all test pairs of 20 × 19 shapes with different resolutions. The best result of each measurement is marked in **bold** font.

Method	Input	Object	Resolution		
			7k	8k	15k
SplineCNN [13]	1	Mesh	27.21	59.01	51.71
ACSCNN [58]	1	Mesh	**0.06**	58.87	53.06
FC [5]	XYZ	Mesh	1.80	53.41	50.04
DiffusionNet [6]	XYZ	Mesh	2.18	6.42	7.28
Multimodal-DFM [51]	XYZ	Mesh	0.51	0.68	0.70
DiffusionNet [6]	WEDS	Mesh	1.79	6.33	12.42
ChebyGCN [12]	WEDS	Mesh	10.88	53.62	57.12
MGCN [15]	WEDS	Mesh	1.14	2.13	6.29
L^2-GNN	WEDS	Mesh	**0.06**	**0.12**	**0.29**

Table 2. Quantitative comparison results of time performance where we perform the dense correspondence on the FAUST dataset. The pre-training and training times are the running times on the CPU and GPU with default parameters for each method, respectively. Only methods that demonstrate robustness to different resolutions are tested.

Method	Pre-processing(s)	Training(s)	Total(s)
DiffusionNet [6]	657.15	2257.67	2914.82
MGCN [15]	3268.19	12213.25	15481.44
Multimodal-DFM [51]	1387.47	3713.33	5100.80
L^2-GNN	**192.37**	**2095.13**	**2287.50**

high accuracy, robustness to resolution and low computational complexity has yet to be developed.

In our network, we incorporate twice linear parameterization to approximate the spectral filter, effectively balancing performance, robustness to variant resolutions, and efficiency. First, the wavelet filter basis functions serve as the foundation of the first linear parameterization, boosting information aggregation by increasing the receptive field without sacrificing locality while providing robustness to variations in mesh discretization and point cloud sampling. Second, with the Chebyshev polynomial basis functions from the second parameterization, we can significantly reduce convolutional complexity. Since we are fitting the wavelet function using the polynomials, the robustness is maintained. To sum up, L^2-GNN provides a fresh solution for 3D deep learning challenges (accuracy, robustness to variant resolutions, and computation complexity), excelling in handling both meshes and point clouds and achieving high performance in tasks like shape correspondence, classification, and segmentation.

4. Experimental Results

In this section, our L^2-GNN blocks provide a plug-and-play framework for different 3D discretization data (mesh and point clouds) and tasks (correspondence, classification, and segmentation). Therefore, we first describe the implementation details of the experiments for different tasks. Then, we provide quantitative and visual comparisons with state-of-the-art methods. Finally, we perform ablation experiments to validate the effectiveness of our L^2-GNN blocks. Our results are obtained using a Core (TM) i9-10900 CPU and a RAM16.0 GB computer. Offline training is conducted on NVIDIA GeForce GTX RTX (24GB memory) GPU.

Table 3. Quantitative comparison results of the mesh classification on SHREC11 dataset.

Method	Input	Object	Accuracy
MeshCNN [3]	XYZ	Mesh	97.50%
Laplacian2Mesh [40]	39D	Mesh	99.50%
FC [5]	XYZ	Mesh	96.67%
DiffusionNet [6]	XYZ	Mesh	99.60%
DiffusionNet [6]	WEDS	Mesh	99.50%
ChebyGCN [12]	WEDS	Mesh	92.33%
MGCN [15]	WEDS	Mesh	99.25%
L^2-GNN	WEDS	Mesh	**99.67%**

4.1. Implementation Details

In this chapter, we describe in detail the experimental procedures and datasets used in the study. To illustrate the hyperparameter configurations in the various experiments, we provide settings for two specific datasets. For the FAUST [61] dataset, we set the learning rate for softmax and hard loss to 0.0005, with weight decay values of 1e-4 and 5e-5 for softmax and hard loss, respectively. In contrast, for the SHREC11 [62] dataset, we used a slightly higher learning rate of 0.01 for softmax, while the learning rate for hard loss remained at 0.0005. The weight decay values for both softmax and hard loss were kept the same at 1e-4 and 5e-5, respectively.

Table 4. Quantitative comparison results of the part segmentation on the SHAPESEG part dataset. The accuracy calculation is based on "hard" ground-truth labels.

Method	Input	Object	Accuracy
MeshCNN [3]	XYZ	Mesh	89.65%
Laplacian2Mesh [40]	39D	Mesh	88.57%
FC [5]	XYZ	Mesh	91.43%
DiffusionNet [6]	XYZ	Mesh	91.21%
DiffusionNet [6]	WEDS	Mesh	90.17%
ChebyGCN [12]	WEDS	Mesh	84.33%
MGCN [15]	WEDS	Mesh	89.25%
L^2-GNN	WEDS	Mesh	**91.55%**

4.1.1 Correspondence

FAUST [61] dataset consists of 100 watertight meshes, including 10 different poses for 10 different subjects with ground-truth corresponding tags. Meanwhile, we use the 80 shapes of the 7k mesh dataset for training and select the last 20 shapes on the 7k, 8k, and 15k resolutions to test the performance. Furthermore, we utilize the nearest neighbor search with the L2 distance in the feature space to find the corresponding point in the test pair of source-target shapes. Finally, we use the average geodesic error of all test shapes to measure the correspondence performance.

4.1.2 Classification

SHREC-11 [62] dataset has 30 categories which have 20 shapes. Similar to other competitors, we only select 10 samples per class for training and the results are the average accuracy of the last 10 epochs of the experiment.

4.1.3 Segmentation

SHAPESEG [63] part dataset contains 381 training shapes from SCAPE [64], FAUST [61], MIT [65], Adobe Fuse [66], and 18 test shapes from SHREC07 [67]. For this task, each point in the 3D shape is classified into one of several predefined component category labels, and the human body

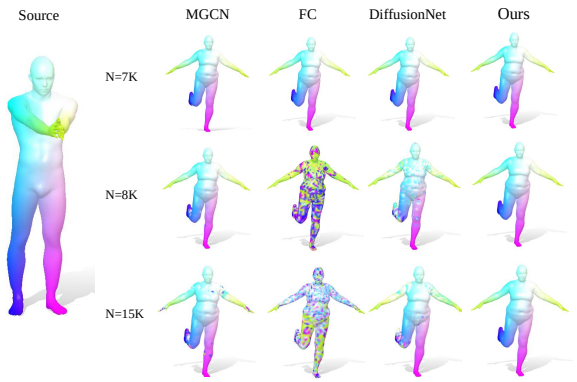


Figure 6. For a selected model, we visualize the feature map extracted by different methods on different resolutions, and the N is the number of vertices. The color distribution closer to the *Source* means a better correspondence result.

is segmented into 8 parts. Finally, we follow the official train/validation/test split scheme of SHAPESEG for our experiments and use the average accuracy (ACC) to evaluate our segmentation performance.

On the other hand, we choose the **Shapenet** part dataset for the part segmentation task of point clouds. This dataset contains 16 object categories and we selected 6 of them for the experiment. Furthermore, we followed the official training/validation/testing split scheme of the Shapenet dataset [68]

4.2. Mesh

4.2.1 Mesh Correspondence

We first apply our L^2-GNN network to the Faust dataset to evaluate the learning performance of shapes correspondence.

For a more intuitive display of the mesh correspondence, we first visualize the correspondence result of FC, MGCN, DiffusionNet, and ours through color transmission, as shown in Fig. 6. Each method has satisfactory results on 7k data since we train them on this resolution. Furthermore, our result is the most consistent with different resolutions when testing on 8k and 15k data, while other methods have clear errors.

Table 1 reports the quantitative comparison results of mesh correspondence on the FAUST dataset. Compared with the SOTA methods, our L^2-GNN achieves the best results of the average geodesic error among different resolutions (7k, 8k, and 15k). Among them, ChebyGCN employs one linear parameterization approach, where its convolution operation is closely tied to data discretization and resolution, which limits its robustness. Furthermore, our results have the smallest error between different resolutions compared to other methods, which is a clear advantage and demonstrates our excellent generalization for the various

Table 5. Quantitative comparison results of point cloud correspondence on FAUST dataset. The calculation of average geodesic error is the same as shape correspondence.

Method	Input	Object	Resolution		
			7k	8k	15k
SplineCNN [13]	1	PointCloud	31.65	53.28	56.94
ChebyGCN [12]	WEDS	PointCloud	21.88	49.37	55.32
ACSCNN [58]	1	PointCloud	5.83	44.26	45.71
FC [5]	XYZ	PointCloud	9.78	57.51	55.44
DiffusionNet [6]	XYZ	PointCloud	4.17	11.59	15.01
Multimodal-DFM [51]	XYZ	PointCloud	0.51	0.73	0.76
DiffusionNet [6]	WEDS	PointCloud	5.89	10.56	16.89
MGCN [15]	WEDS	PointCloud	11.34	15.27	16.61
L^2-GNN	WEDS	PointCloud	**0.06**	**0.14**	**0.31**

degrees of discrete 3D data.

Finally, we compare the time consumption of the corresponding task with the SOTA methods robust to varying resolutions on the FAUST dataset. Generally, a GCN network can be divided into data processing and training. Therefore, we count the pre-training, training, and total time separately in Table 2.

We can observe that the pre-processing time of FC is large because the method needs to calculate geodesic distances. DiffusionNet is a good competitor in terms of time. Their pre-processing time is higher than ours due to the calculation of eigenvector matrices and eigenvalues matrices. Multimodal-DFM ranks as the second most time-consuming method, primarily due to its incorporation of multimodal data. Furthermore, MGCN is the most time-consuming since its complex computation for eigenvector and eigenvalues in pre-processing and the wavelet matrix multiplication for each epoch in training. Our method requires only a brief pre-computation step, which is more efficient and does not significantly impact overall performance. As a result, our method is significantly faster than other methods due to our efficient spectral filter.

4.2.2 Mesh Classification

Here, we conduct mesh classification comparison on SHREC-11 [62] dataset as shown in Table 3. Due to the small scale of the classification task, we first randomly select 8 out of 32 scales for filters and utilize two L^2-GNNOV layers to construct our classification network. Then we train 60 epochs and converge quickly at 30 epochs. Nonetheless, our L^2-GNN achieves competitive results compared to the current SOTA networks (such as MeshCNN, FC, DiffusionNet, etc.).

4.2.3 Mesh Segmentation

Finally, we extended our L^2-GNN model to the part segmentation task on the SHAPESEG [63] dataset. Different

Table 6. Quantitative comparison results of the point cloud classification on SHREC11 dataset.

Method	Input	Object	Accuracy
PointNet++ [8]	XYZ	PointCloud	90.25%
DGCNN [9]	XYZ	PointCloud	95.67%
Pointnext [10]	XYZ	PointCloud	99.25%
DiffusionNet [6]	XYZ	PointCloud	**99.67%**
DiffusionNet [6]	WEDS	PointCloud	99.25%
ChebyGCN [12]	WEDS	PointCloud	90.12%
MGCN [15]	WEDS	PointCloud	96.25%
L^2-GNN	WEDS	PointCloud	**99.67%**

Table 7. Quantitative comparison results of the part segmentation on the ShapeNet part dataset. We show averaged accuracy in % and select six representative objects to compare with five different network structures.

	Earphone	Pistol	Table	Car	Guitar	Skateboard	Average
#Shapes	55	239	4423	740	628	121	
PointNet++ [8]	80.63	70.24	79.21	72.71	74.52	85.88	77.20
DGCNN [9]	87.20	91.86	88.15	89.25	**95.20**	94.22	90.98
Pointnext [10]	87.68	92.02	87.28	90.24	94.24	**94.95**	91.07
DiffusionNet-XYZ [6]	86.50	91.45	**88.55**	87.59	92.56	92.55	89.87
DiffusionNet-WEDS [6]	86.55	91.95	87.65	90.10	93.38	91.79	90.23
ChebyGCN [12]	80.23	85.67	80.17	87.19	79.25	88.35	84.46
MGCN [15]	82.23	89.52	87.01	85.25	85.16	90.26	86.57
L^2-GNN	**88.20**	**92.78**	87.26	**90.75**	94.12	94.64	**91.29**

from the classification structure, we randomly select 16 filters in 32 different scales and use two FMGCONV layers after a two-layer MLP. After that, we use two shared full-connection layers (64, 64) to transform the features. Table 4 reports the evaluation results compared to Laplacian2Mesh, DiffusionNet, and other methods. We also achieve SOTA segmentation results and a 2.3% improvement over our baseline MGCN while maintaining fast training. Furthermore, we visualize the human segmentation results of FC, MGCN, DiffusionNet, and ours, as shown in Fig. 7. Our method achieves the most satisfactory segmentation results, such as the thigh in various human poses.

4.3. Point Cloud

4.3.1 Point Cloud Correspondence

For the correspondence task of the point cloud, we only use the coordinate information in the FAUST dataset and calculate the Laplacian matrix by the existing robust-laplacian function. Meanwhile, we also use the comparison methods in mesh correspondence for point clouds. Table 5 reports the overall comparison results. Our network also achieves state-of-the-art results at different resolutions of the point cloud.

Figure 7. Visualization comparisons of mesh segmentation on human between our method and state-of-the-art methods.

4.3.2 Point Cloud Classification

Similarly, we use the coordinate information of the SHREC11 dataset for the point cloud classification. We further select some point cloud classification methods such as PointNet++, DGCNN, and Pointnext to expand the comparison. The comparison results are shown in the Table 6. Our L^2-GNN achieves competitive classification results, which are consistent with the mesh classification and further demonstrate the performance of our network in handling different types of 3D data. In addition, SHREC11 has some sharp shapes and few points, which leads to the failure of generating the Laplacian matrix on the point cloud data. Therefore, we increase the n-neighbors parameter of the intrinsic Laplacian function. From the result, we can observe our L^2-GNN and DiffusionNet achieve the best performance in the task point cloud classification.

4.3.3 Point Cloud Segmentation

Here, we conduct the point cloud segmentation on the Shapenet part dataset while our network structure is simi-

Table 8. Ablation experiment of the filter numbers on the network model of the corresponding task.

	K	24	16	8	4
MGCN	error	0.923	1.146	1.379	2.425
	timer(s)	22168.13	15481.44	8975.34	6060.17
L^2-GNN	error	0.052	0.061	0.271	1.178
	timer(s)	2645.53	2287.50	1927.48	1650.49

lar to our mesh segmentation structure. Then, we compared different point cloud segmentation methods, and Table 7 reports the quantitative results. We achieve SOTA results in some shapes, achieved competitive in others, and attained the best outcomes in terms of average accuracy. Furthermore, we visualized some point cloud segmentation results as shown in Fig. 8 while our results are closer to GT, especially in some details.

4.4. Ablation Study

We introduce a novel method for approximating spectral filters through a twice-linear parameterization approach. We first parameterize the spectral filters using wavelet filter basis functions, and then further parameterize these wavelet basis functions using Chebyshev polynomial basis functions. If the first parameterization is omitted, our ablation study compares the approach with spectral methods like ChebyGCN [12]. On the other hand, if the second parameterization is omitted, the comparison in the ablation study is made with MGCN [15].In the experimental section, we provide comparisons that help validate the contribution of each component to the overall performance of L^2-GNN.

We conducted the following three ablation experiments to demonstrate the robustness of our method. First, we select multiple wavelet filters with different scales to capture low-frequency and high-frequency features for comparison with MGCN. Then, we added two additional levels of noise to the dataset to compare with other methods. Moreover, we compare the performance results of multiple methods on the inputs of WEDS and XYZ. Finally, we further explored the learnable coefficients for the basis functions.

Filter numbers. We conduct an ablation experiment of the filter numbers on our correspondence network model. Meanwhile, MGCN is the baseline of our network and we choose it for performance reference.

As shown in Table 4.3.2, we select four different numbers of filters from 32 scales for training to compare their training time and performance on the FAUST dataset. We can observe that the performance of our L^2-GNN and MGCN improves with the K increases. However, the training time of MGCN grows squarely with K increasing, since their additional calculation of multiple wavelet filter wavelet basis on each shape. Meanwhile, the training time of our L^2-GNN increases linearly with K due to the twice linear parameterization. Finally, we select K=16 as the default filter number in our L^2-GNN to achieve the balance between performance and training time.

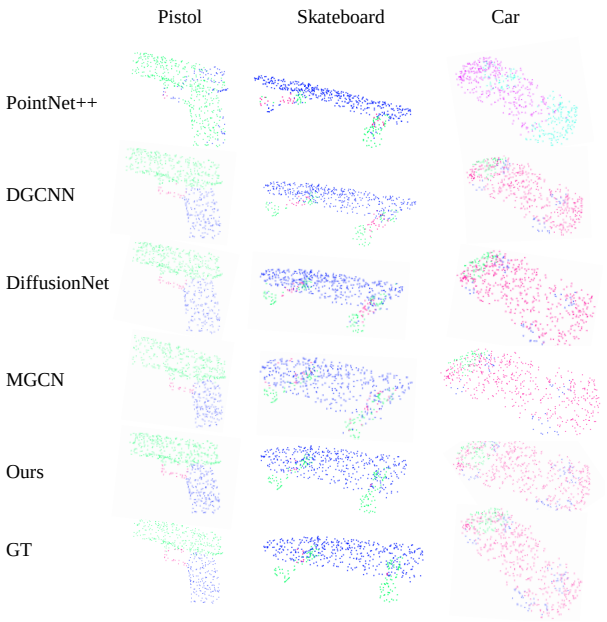


Figure 8. Visualization comparisons of point cloud segmentation on diverse objects between our method and state-of-the-art methods.

Table 9. Ablation experiment of different noise data on the network model of the corresponding task. Each method are employed with optimal configurations.

Method	No-noise	0.2%	0.3%
SplineCNN [13]	27.21	28.33	28.46
ACSCNN [58]	0.06	2.98	3.40
FC [5]	1.80	1.45	2.56
Diffusionnet [6]	2.18	3.25	5.43
ChebyGCN [12]	10.88	14.57	13.48
MGCN [15]	1.14	3.45	4.27
L^2-GNN	**0.06**	**0.45**	**0.83**

Table 10. Ablation experiment of different input features on the network model of the corresponding task

Method	WEDS	XYZ
Diffusionnet [6]	1.79	2.18
ChebyGCN [12]	10.62	10.88
MGCN [15]	1.14	3.25
L^2-GNN	0.06	0,09

Noise Data. Furthermore, we introduce random Gaussian noise to the mesh vertices for the 7K mesh correspondence task. Specifically, we add noise with a standard deviation of 0.2% and 0.3% of the length of the bounding box diagonal. As demonstrated in Table 9, our L^2-GNN consis-

tently outperforms state-of-the-art (SOTA) methods, further validating the robustness of our proposed L^2-GNN. The results show that our model is not only effective in handling clean data but also showcases its superiority in maintaining high performance even in the presence of noise.

Different Input. Moreover, we chose to compare the effects of inputting XYZ and WEDS on the mesh dataset. For mesh, in addition to the vertex coordinate information, the connectivity information within the mesh is crucial. WEDS incorporates not only the coordinate information P but also the connectivity information. As shown in Table 10, in the case of a mesh, using WEDS as input yields better results compared to directly inputting vertex coordinate information. Therefore, for mesh data, we chose to use WEDS as the input. It is worth noting that even when using coordinate information as input, our method still outperforms other approaches.

5. Conclusions

In this paper, we propose the L^2-GNN, which uses twice linear parameterization to approximate the spectral filter of the graph neural network. We achieve both performance and discretization robustness goals simultaneously through the following two aspects. On the one hand, we use wavelet filter basis functions to parameterize the spectral filter, which allows the network to capture both low-frequency and high-frequency information thus further improving its performance. On the other hand, we use the Chebyshev polynomial basis functions to parameterize the wavelet filter basis functions, which reduces the computational complexity of graph convolution while maintaining the robustness of different discretizations and samplings. Our L^2-GNN exhibits strong adaptability to both meshes and point clouds and demonstrates impressive performance on tasks such as shape correspondence, classification, and segmentation. One limitation of our method is that it still requires pre-computing the coefficients for approximating wavelet filter basis functions with Chebyshev basis functions. We consider exploring alternative basis functions in the spectral domain that may offer improved performance compared to the current wavelet filter basis functions. One promising way is using learnable basis functions, which have the potential to capture complex relationships within the data. Although initial attempts to replace wavelet basis functions with learnable ones did not yield satisfactory results in the supplemental materials, further research is needed to better understand the challenges and opportunities of this approach.

Acknowledgements

This work was supported by the National Key R&D Program of China (2022ZD0160804) and the National Natural Science Foundation of China (62202076, 52238003, 62306297, and 62206137).

References

[1] 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 workshops*, pages 37–45, 2015. 1, 2

[2] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. *Advances in neural information processing systems*, 29, 2016. 1, 2

[3] 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. 1, 2, 7

[4] Nicolas Donati, Abhishek Sharma, and Maks Ovsjanikov. Deep geometric functional maps: Robust feature learning for shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8592–8601, 2020. 1

[5] Thomas W Mitchel, Vladimir G Kim, and Michael Kazhdan. Field convolutions for surface cnns. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10001–10011, 2021. 1, 2, 3, 6, 7, 8, 10

[6] 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. 1, 3, 6, 7, 8, 9, 10

[7] 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, 3

[8] 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, 3, 9

[9] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 1, 3, 9

[10] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 35:23192–23204, 2022. 1, 3, 9

[11] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 1, 3

[12] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016. 1, 3, 4, 5, 6, 7, 8, 9, 10

[13] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 869–877, 2018. 1, 2, 3, 6, 8, 10

[14] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. Graph wavelet neural network. *arXiv preprint arXiv:1904.07785*, 2019. 1, 3

[15] Yiqun Wang, Jing Ren, Dong-Ming Yan, Jianwei Guo, Xiaopeng Zhang, and Peter Wonka. Mgcn: descriptor learning using multiscale gcns. *ACM Transactions on Graphics (TOG)*, 39(4):122–1, 2020. 1, 3, 6, 7, 8, 9, 10

[16] Mohammad Farazi, Wenhui Zhu, Zhangsihao Yang, and Yalin Wang. Anisotropic multi-scale graph convolutional network for dense shape correspondence. *arXiv preprint arXiv:2210.09466*, 2022. 1, 3

[17] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011. 1, 3, 5

[18] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part III 11*, pages 356–369. Springer, 2010. 2

[19] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009. 2

[20] Hanyu Wang, Jianwei Guo, Dong-Ming Yan, Weize Quan, and Xiaopeng Zhang. Learning 3d keypoint descriptors for non-rigid shape matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. 2

[21] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 1626–1633. IEEE, 2011. 2

[22] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational projection statistics for 3d local surface description and object recognition. *International journal of computer vision*, 105:63–86, 2013. 2

[23] Yiqun Wang, Jianwei Guo, Jun Xiao, and Dong-Ming Yan. A wavelet energy decomposition signature for robust non-rigid shape matching. In *SIGGRAPH Asia 2019 Posters*, pages 1–2. 2019. 2

[24] Yi Ding, Neethu Robinson, Chengxuan Tong, Qiuhao Zeng, and Cuntai Guan. Lggnet: Learning from local-global-graph representations for brain–computer interface. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. 2

[25] Mianxin Liu, Han Zhang, Feng Shi, and Dinggang Shen. Hierarchical graph convolutional network built by multiscale atlases for brain disorder diagnosis using functional connectivity. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. 2

[26] 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. 2

[27] Shuyi Niu, Junxiao Sun, Youyong Kong, and Huazhong Shu. Gcn2caps: Graph convolutional network to capsule network for wide-field robust graph learning. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 2219–2223. IEEE, 2022. 2

[28] Pim De Haan, Maurice Weiler, Taco Cohen, and Max Welling. Gauge equivariant mesh cnns: Anisotropic convolutions on geometric graphs. *arXiv preprint arXiv:2003.05425*, 2020. 2

[29] Wenchong He, Zhe Jiang, Chengming Zhang, and Arpan Man Sainju. Curvanet: Geometric deep learning based on directional curvature for 3d shape analysis. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2214–2224, 2020. 2

[30] Adrien Poulenard and Maks Ovsjanikov. Multi-directional geodesic neural networks via equivariant convolution. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018. 2

[31] Chang-Qin Huang, Fan Jiang, Qiong-Hao Huang, Xi-Zhe Wang, Zhong-Mei Han, and Wei-Yu Huang. Dual-graph attention convolution network for 3-d point cloud classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 2

[32] Zhangsihao Yang, Or Litany, Tolga Birdal, Srinath Sridhar, and Leonidas Guibas. Continuous geodesic convolutions for learning on 3d shapes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 134–144, 2021. 2

[33] Ruben Wiersma, Elmar Eisemann, and Klaus Hildebrandt. Cnns on surfaces using rotation-equivariant features. *ACM Transactions on Graphics (ToG)*, 39(4):92–1, 2020. 2

[34] Dominique Beaini, Saro Passaro, Vincent Létourneau, Will Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. In *International Conference on Machine Learning*, pages 748–758. PMLR, 2021. 2

[35] Bi'an Du, Xiang Gao, Wei Hu, and Xin Li. Self-contrastive learning with hard negative sampling for self-supervised point cloud learning. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 3133–3142, 2021. 2

[36] Zeyu Cui, Zekun Li, Shu Wu, Xiaoyu Zhang, Qiang Liu, Liang Wang, and Mengmeng Ai. Dygcn: Efficient dynamic graph embedding with graph convolutional network. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 2

[37] Yue Wu, Jiaming Liu, Maoguo Gong, Peiran Gong, Xiaolong Fan, AK Qin, Qiguang Miao, and Wenping Ma. Self-supervised intra-modal and cross-modal contrastive learning for point cloud understanding. *IEEE Transactions on Multimedia*, 2023. 2

[38] Xuebin Zheng, Bingxin Zhou, Junbin Gao, Yu Guang Wang, Pietro Lio, Ming Li, and Guido Montúfar. How framelets enhance graph neural networks. *arXiv preprint arXiv:2102.06986*, 2021. 2

[39] Mengxi Yang, Xuebin Zheng, Jie Yin, and Junbin Gao. Quasi-framelets: Another improvement to graphneural networks. *arXiv preprint arXiv:2201.04728*, 2022. 2

[40] 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. 2, 7

[41] 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. 3

[42] Zhongpai Gao, Junchi Yan, Guangtao Zhai, Juyong Zhang, and Xiaokang Yang. Robust mesh representation learning via efficient local structure-aware anisotropic convolution. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. 3

[43] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018. 3

[44] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019. 3

[45] Lukas Bernreiter, Lionel Ott, Roland Siegwart, and Cesar Cadena. Sphnet: A spherical network for semantic pointcloud segmentation. *arXiv preprint arXiv:2210.13992*, 2022. 3

[46] Adrien Njanko and Danda B Rawat. On the identification of isomorphic graphs for graph neural network using multi-graph approach. In *2022 IEEE 13th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0061–0066. IEEE, 2022. 3

[47] Jing Yi and Zhenzhong Chen. Multi-modal variational graph auto-encoder for recommendation systems. *IEEE Transactions on Multimedia*, 24:1067–1079, 2021. 3

[48] Xiang Ling, Lingfei Wu, Saizhuo Wang, Tengfei Ma, Fangli Xu, Alex X Liu, Chunming Wu, and Shouling Ji. Multilevel graph matching networks for deep graph similarity learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 3

[49] Mingze Sun, Shiwei Mao, Puhua Jiang, Maks Ovsjanikov, and Ruqi Huang. Spatially and spectrally consistent deep functional maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14497–14507, 2023. 3

[50] Dongliang Cao, Paul Roetzer, and Florian Bernard. Unsupervised learning of robust spectral shape matching. *arXiv preprint arXiv:2304.14419*, 2023. 3

[51] Dongliang Cao and Florian Bernard. Self-supervised learning for multimodal non-rigid 3d shape matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17735–17744, 2023. 3, 6, 8

[52] Bharti Khemani, Shruti Patil, Ketan Kotecha, and Sudeep Tanwar. A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions. *Journal of Big Data*, 11(1):18, 2024. 3

[53] Yuyol Shin and Yoonjin Yoon. Pgcn: Progressive graph convolutional networks for spatial–temporal traffic forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 2024. 3

[54] Guan-Qiang Wang, Chi-Zhou Zhang, Ming-Song Chen, Yong-Cheng Lin, Xian-Hua Tan, Yu-Xin Kang, Qiu Wang, Wei-Dong Zeng, and Wei-Wei Zhao. A high-accuracy and lightweight detector based on a graph convolution network for strip surface defect detection. *Advanced Engineering Informatics*, 59:102280, 2024. 3

[55] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 3

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

[57] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1416–1424, 2018. 3

[58] Qinsong Li, Shengjun Liu, Ling Hu, and Xinru Liu. Shape correspondence using anisotropic chebyshev spectral cnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14658–14667, 2020. 3, 6, 8, 10

[59] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 3

[60] Nicholas Sharp and Keenan Crane. A laplacian for nonmanifold triangle meshes. In *Computer Graphics Forum*, volume 39, pages 69–80. Wiley Online Library, 2020. 5

[61] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3794–3801, 2014. 7

[62] Yunhai Wang, Shmulik Asafi, Oliver Van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG)*, 31(6):1–10, 2012. 7, 8

[63] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.*, 36(4):71–1, 2017. 7, 8

[64] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *ACM SIGGRAPH 2005 Papers*, pages 408–416. 2005. 7

[65] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. In *ACM SIGGRAPH 2008 papers*, pages 1–9. 2008. 7

[66] Adobe Mixamo. Animate 3d characters for games, film, and more, 2021. 7

[67] Daniela Giorgi, Silvia Biasotti, and Laura Paraboschi. Shape retrieval contest 2007: Watertight models track. *SHREC competition*, 8(7):7, 2007. 7

[68] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 8