IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 2, 2020, accepted February 16, 2020, date of publication February 19, 2020, date of current version February 28, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2975067

A Comprehensive Survey on Geometric Deep Learning

WENMING CAO [1,2,3], (Member, IEEE), ZHIYUE YAN [1,2], ZHIQUAN HE [1,2], AND ZHIHAI HE [3], (Fellow, IEEE)

[1]The Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen 518060, China
[2]Guangdong Multimedia Information Service Engineering Technology Research Center, Shenzhen 518060, China
[3]Video Processing and Communication Laboratory, Department of Electrical and Computer Engineering, University of Missouri, Columbia, MO 65211, USA

Corresponding author: Zhihai He (hezhi@missouri.edu)

This work was supported in part by the National Natural Science Foundation of China under Grant 61771322, Grant 61871186, and Grant 61971290.

ABSTRACT Deep learning methods have achieved great success in analyzing traditional data such as texts, sounds, images and videos. More and more research works are carrying out to extend standard deep learning technologies to geometric data such as point cloud or voxel grid of 3D objects, real life networks such as social and citation network. Many methods have been proposed in the research area. In this work, we aim to provide a comprehensive survey of geometric deep learning and related methods. First, we introduce the relevant knowledge and history of geometric deep learning field as well as the theoretical background. In the method part, we review different graph network models for graphs and manifold data. Besides, practical applications of these methods, datasets currently available in different research area and the problems and challenges are also summarized.

INDEX TERMS Convolutional neural networks, geometric deep learning, graph, manifold.

I. INTRODUCTION

A. INTRODUCTION

The deep Learning [1] technologies, for example, the convolutional neural networks [2], have achieved unprecedented good results in some machine learning applications such as object detection [3]–[5], image classification [6], speech recognition [7], and machine translation [8]. Different from traditional neural networks, the deep neural networks, especially convolutional neural networks, make use of the basic statistical characteristics of data including local stationarity and multi-scale component structure to capture deeper local information and features. Although deep learning technology is very successful in processing traditional signals such as image, sound, video or text, the current research on deep learning still mainly focuses on the data mentioned above which are defined in the Euclidean domain, namely grid-like data. With the emergence of larger data scale and more powerful GPU computing ability, people begin to be more and more interested in processing data in non-Euclidean domain, such as graphs and manifolds. This type of data is ubiquitous in real life. It is of great significance to study deep learning techniques in non-Euclidean domains. This is called geometric deep learning.

The geometric deep learning mainly study graph and manifold data. The graph is composed of nodes and edges of the network structure data. For instance, in social network, each node represent a person's information and the edge represent the relationship between people. The edges can be directed or undirected depending on the relationship of the connecting vertices. The Manifold data are usually used to describe geometric shapes, such as surface of objects returned by radar scanning. These geometric data are irregularly arranged and randomly distributed, which makes it difficult for people to find out the underlying pattern. Specifically, it is difficult to find the neighbor nodes of a certain point in the data, or the number of a node's neighbor is different in [9]. This makes it difficult to define convolution operations like those on images. On the other hand, data like images in the Euclidean domain can be regarded as a special graph data, with vertices arranged in a regular way. Another issue is that non-Euclidean data usually has extraordinarily large scale. For example, molecular graph can have hundreds of millions of nodes. For this case, it is unlikely to use the traditional deep learning

The associate editor coordinating the review of this manuscript and approving it for publication was Guitao Cao.

technology to carry out analysis and prediction tasks. This is why deep learning is so important in the field of geometric data.

The purpose of this survey is to review and summarize the geometric deep learning frameworks and algorithms developed on graphs and manifolds data, and to introduce the practical difficulties and development directions in this new rising field.

B. BRIEF HISTORY

The history of geometric deep learning filed is not very long. Although it seems that this area is in its infancy, the deep learning behind its development has a long history. The rapid development of deep learning technologies such as convolutional neural network greatly pushes the progress of geometric deep learning. After all, studying geometric deep learning is to study how to define the non-Euclidean world mathematically, and how to transplant the existing frameworks and algorithms of deep learning to handle graphs and manifolds data effectively. In general, geometric deep learning can be mainly divided into two major research directions, one is for processing graph data (*i.e.* graph networks or grid-like data); the other is for processing manifold data (*i.e.* generally for processing 3D point cloud data). Among them, graph data processing is more popular, and people aim to extend the deep learning technologies to non-Euclidean structural data.

As early as in 2005, M. Gori *et al.* first proposed a graph neural network (GNN) to process graph data [10] such as directed graphs, undirected graphs, labeled graphs, and recurrent graphs. The work of [11] published by Scarselli *et al.* in 2009 brought back the graph neural network model to the public's horizon, defined a function that can map graph and any node to a dimensional Euclidean space, and proposed an algorithm to estimate the neural network model parameter with supervised learning. In the work of [12] proposed spectral convolutional neural networks on graphs. Work of [13] extended the spectral network by combining a graph estimation process. diffusion convolutional neural network (DCNN) was next proposed in [14] to learning diffusion based representation from graph data for node classification. The work of [9], similar to image based convolutional network operating on the input locally connected region, proposed a general method to extract the locally connected region from the graph. In 2016, M. Defferrard *et al.* proposed ChebNet [15], and then a simplified version GCN (graph convolutional network) was proposed [16]. One year later, CayleyNet was proposed by Levie *et al.* [17]. All the above research results were based on the idea of convolutional network. Besides graph convolution model, there are other similar studies conducted in parallel, such as graph attention networks, graph generative networks, and graph auto-encoders that will be seen in section III.

At the same time, research of deep learning theory on manifold data are also carried out. There have been two traditional research methods on manifolds, one is to fill 3D shapes with many voxel grids (cube blocks), and each voxel can be processed by 3D CNN opreation, called 3D volumetric CNN. The other is to take photos of 3D objects from multiple angles to increase the data source of the same object, which is called multi-view CNN. In 2015, J. Masci *et al.* proposed the framework of Geodesic CNN [18], which is the promotion of convolutional neural network (CNN) paradigm on non-Euclidean manifold. Later on, in the work [19], the authors proposed Anisotropic CNN framework in the study of intrinsic dense correspondences between deformable shapes on the experiment of the results over [18]. This method generalized convolutional neural networks to the non-Euclidean domain by replacing the traditional convolution operations by projections on a set of oriented anisotropic diffusion kernels. Related work is [20], which proposed SyncSpecCNN network, where the kernel is parameterized in the spectral domain spanned by the Laplacian feature basis. D. Boscaini *et al.* proposed Localized Spectral CNN (LSCNN) in [21], in which the model structure is based on local frequency analysis with a windowed Fourie transform to manifold data. This method can be used for deformable shapes. A new framework called FMNet was introduced to learn the dense correspondence between deformable 3D shapes [22].

Many works have been conducted to find better approach to generalize convolution-like operations of convolution neural networks to the non-Euclidean domain. For example, the work [23] proposed a unified CNN framework MoNet and declared that the previous various CNN models can be unified within the framework. In addition, many researchers have tried to apply the above methods to a wide range of practical problems, from biochemistry [24] and skeleton-based human motion recognition task [25] to the recommender systems [26].

C. ORGANIZATION OF THIS SURVEY

The rest of this survey is organized in the following. Section II gives some basic background knowledge and conceptions of geometric deep learning. Section III overviews the typical deep learning approaches on graphs and manifolds; Section IV presents a gallery of applications across various tasks and introduces some common datasets. Section V discusses the current challenges and future directions and section VI summaries this work.

D. RELATED SURVEYS AND OUR CONTRIBUTIONS

There has been some existing professional reviews in the geometric deep learning field. Bronstein *et al.* gave an overview of deep learning methods in the non-Euclidean domain, including processing methods on graphs and manifolds [27]. In addition, this review also provided a complete mathematical derivation and various formulas in the field of geometric deep learning, which serves a good reference for new starters. However, the research of graph deep learning is going fast. After this survey, may new research results have been achieved, including graph attention networks, graph auto-encoders and other architectures, and some new

benchmarks. The work of [28], [29] put forward a most up-to-date survey on deep learning for graphs, which partially updated the work in [27]. Still, it did not cover those studies on graph generative and graph attention networks and methods on manifolds. Most recently, based on the above two surveys, Wu *et al*. put forward the current network structures on graphs, including spatio-temporal networks [30]. However, the background knowledge part was only simply mentioned based on the previous theoretical introductions, which is not friendly for new readers.

This work serves as a comprehensive survey that covers the latest deep learning methods for graphs and manifolds data, and the related applications. In the introduction part, we try our best to use plain language and some simple mathematical derivations to explain the theory of geometric deep learning. In the method part, we summary the network architectures and methods for graph and manifold data, as well as the spatial and spectral-based methods. We also include many latest existing research results, such as the hot point cloud technology in the recent two years. In the end part of this review, we point out some limitations of existing algorithms and the possible development directions of this field in the future. We believe that readers, especially new beginners, will have a comprehensive and clear understanding of geometric deep learning after reading this review.

II. BACKGROUND

In this section, we give a basic background introduction on graph and manifold theories.

A. GRAPH AND LAPLACIAN MATRIX

The definition of graph is $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ where $\mathbf{V}$ is the set of vertices or nodes, $\mathbf{E}$ is the set of edges. Let $v_i \in \mathbf{V}$ represent a node and $e_{ij} = (v_i, v_j) \in \mathbf{E}$ to represent an edge between v_i and v_j. Let $N = |\mathbf{V}|$ to denote the total number of nodes and $M = |\mathbf{E}|$ to denote the total number of edges. The adjacency matrix $\mathbf{A}$ is an $N \times N$ matrix of edge weights where:

$$\mathbf{A}_{ij} = a_{ij} > 0 if\ e_{ij} \in \mathbf{E} \tag{1}$$

$$\mathbf{A}_{ij} = 0 if\ e_{ij} \notin \mathbf{E} \tag{2}$$

The edges in an undirected graph are all undirected, *i.e.* an undirected graph's edges are all undirected, any two nodes are just connected without direction. For an undirected graph, $\mathbf{A}_{ij} = \mathbf{A}_{ji} = 1$. A directed graph has edges going from one node to another. For a directed graph, $\mathbf{A}_{ij} \neq \mathbf{A}_{ji}$.

The Laplacian matrix of a graph is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the degree matrix, which is diagonal. The diagonal elements are the degrees of nodes, which is defined as the number of connecting edges, namely, $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$). The common Laplacian matrix usually has the following three forms, Combinatorial Laplacian:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \tag{3}$$

Symmetric Normalized Laplacian:

$$\mathbf{L}^{sym} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \tag{4}$$

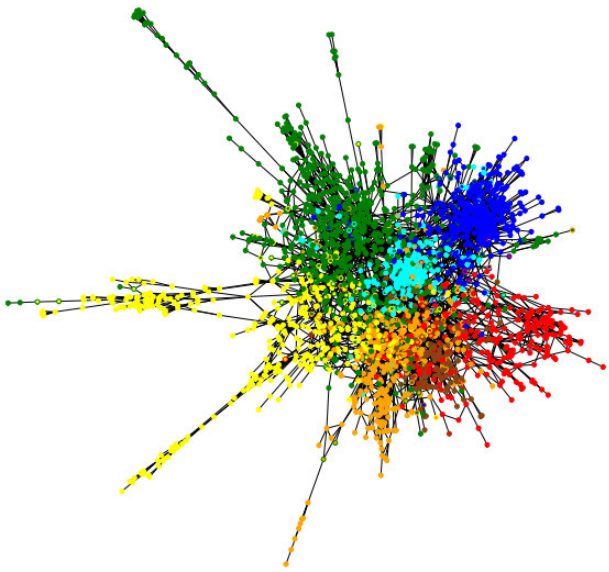

FIGURE 1. Example graph network of Cora dataset. Marker fill color represents the predicted class, marker outline color represents the ground truth class. Cited from [23].

and Random Walk Normalized Laplacian:

$$\mathbf{L}^{rw} = \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A} \tag{5}$$

For an undirected graph G, the Laplacian matrix $\mathbf{L}$ is symmetric and positive-semidefinite. The dimension of the eigenspace is N.

Since we assumed that G is a simple graph, its adjacency matrix $\mathbf{A}$ only contains 1s or 0s and its diagonal elements are all 0s. So the elements of $\mathbf{L}$ are given by:

$$\mathbf{L}_{i,j} = \begin{cases} \deg(v_i) & \text{if } i = j, \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

where $deg(v_i)$ is the degree of the vertex i. The elements of $\mathbf{L}^{sym}$ are given by:

$$\mathbf{L}^{sym}_{i,j} = \begin{cases} 1 & \text{if } i = j \text{ and } \deg(v_i) \neq 0, \\ \frac{-1}{\sqrt{deg(v_i)deg(v_j)}} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

where the eigenvalues (known as the spectrum of the normalized Laplacian) of the normalized symmetric Laplacian satisfy $0 \leq \lambda_0 \leq \cdots \leq \lambda_{N-1} \leq 2$. Next the elements of $\mathbf{L}^{rw}$ are given by:

$$\mathbf{L}^{sym}_{i,j} = \begin{cases} 1 & \text{if } i = j \text{ and } \deg(v_i) \neq 0, \\ \frac{-1}{deg(v_i)} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

It is important to note that the Laplacian matrix of different forms above is nothing more than using degree matrix with different normalization strategies.

B. FOURIER TRANSFORM ON GRAPHS

The key of graph convolutional network based on spectral method is the eigen-decomposition of Laplacian matrix, so Laplacian matrix plays a very important role. The normalized Laplacian matrix is a semi-positive definite symmetric matrix, it has N linearly independent and mutually orthogonal eigenvectors, and N non-negative eigenvalues. Therefore, Laplacian matrix can be decomposed into the following formula:

$$\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^{-1} \tag{9}$$

where $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \cdots, \mathbf{u}_{N-1}] \in \mathbf{R}^{N\times N}$ is a matrix composed of corresponding eigenvectors sorted by eigenvalues. We know that $\mathbf{U}$ is an orthogonal matrix and has $\mathbf{U}^{-1} = \mathbf{U}^{\mathrm{T}}$, these eigenvectors form an orthogonal space; Λ is a diagonal matrix of eigenvalues and has $\Lambda_{ii} = \lambda_i$, $0 = \lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_{N-1}$. In the N-dimensional space determined by graph, the smaller eigenvalue λ_i means the corresponding eigenvector $\mathbf{u}_i$ has less importance, and can be even ignored. The Laplacian eigen-decomposition can also be written as:

$$\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^{\mathrm{T}} \tag{10}$$

Since the Laplacian matrix is a discrete Laplacian operator, it is natural to use the Laplacian matrix and its eigenvectors to define the Fourier transform of the graph in spectral method. In order to transfer the traditional Fourier transform and convolution operation to graph, the core work is to transform the eigenfunction (basis function) $e^{-i\omega t}$ of the Laplacian operator into the eigenvector of the corresponding Laplacian matrix on the graph. The relevant work [31] has made outstanding contributions and adequate explanations about this theory. In graph signal processing, a graph signal $\mathbf{x} \in \mathbf{R}^N$ is a feature vector of the nodes of a graph where $\mathbf{x}_i$ is the value of the i^{th} node. So the graph Fourier transform to a signal $\mathbf{x}$ is defined as $\hat{\mathbf{x}} = \mathbf{U}^{\mathrm{T}}\mathbf{x}$ and the inverse graph Fourier transform is defined as $\mathbf{x} = \mathbf{U}\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ represents the resulting signal from graph Fourier transform. The graph Fourier transform projects the input graph signal to the orthonormal space where the basis is formed by eigenvectors of the normalized graph Laplacian.

C. CONVOLUTION ON GRAPHS

The convolution theorem states that the Fourier transform of convolution is the product of Fourier transforms, by analogy to the graph and puts in the definition of the Fourier transform on graph. Now the graph convolution of the input signal $\mathbf{x}$ with a filter $\mathbf{g} \in \mathbf{R}^N$ is defined as:

$$\mathbf{x} *_G \mathbf{g} = \mathbf{U}\left(\left(\mathbf{U}^{\mathrm{T}}\mathbf{x}\right) \odot \left(\mathbf{U}^{\mathrm{T}}\mathbf{g}\right)\right) \tag{11}$$

where $\odot$ denotes the Hadamard product, in which the two elements from the sample location are multiplied. Use $\mathbf{g}_\theta = \mathrm{diag}\left(\mathbf{U}^{\mathrm{T}}\mathbf{g}\right)$ to denote as a filter in spectral domain, then the graph convolution is simplified as:

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U} \cdot \mathbf{g}_\theta \left(\mathbf{U}^{\mathrm{T}}\mathbf{x}\right) \tag{12}$$

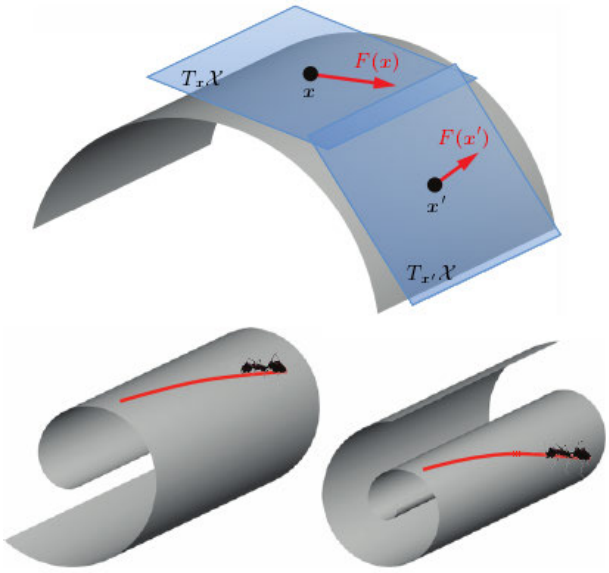

FIGURE 2. Top: tangent space and tangent vectors on a two-dimensional manifold (surface). Bottom: Examples of isometric deformations. Cited from [27].

The Spectral-based graph convolutional networks all follow this definition, and the most critical difference is the selection of filter $\mathbf{g}_\theta$.

D. MANIFOLD

1) MANIFOLD BASICS

From the wikipeida, a manifold is a topological space that locally resembles Euclidean space near each point. More precisely, each point of an n-dimensional manifold has a neighborhood that is homeomorphic to the Euclidean space of dimension n. Globally, it may be not homeomorphic to Euclidean space. The surface of a sphere is such an example. Formally speaking, let X be a d-dimensional differentiable manifold with a boundary ∂X. Around a point $x \in$ X, there is a neighborhood that is topologically equivalent (homeomorphic) to a d-dimensional Euclidean space referred to as the tangent space and denoted by T_xX. On each tangent space, an inner product $\langle\cdot,\cdot\rangle_{T_x\mathrm{X}} : T_x\mathrm{X} \times T_x\mathrm{X} \to \mathbb{R}$ depending smoothly on x is called the Riemannian metric in differential geometry and allows performing local measurements of angles, distances, and volumes. A manifold equipped with a metric is called a Riemannian manifold, which is completely abstract. In shape analysis, two-dimensional manifolds (surfaces) embedded into $\mathbb{R}^3$ are used in computer graphics and vision to describe the boundary surfaces of 3D objects, commonly known as ''3D shapes''. This term is a bit misleading, because ''3D'' here refers to the dimensionality of the embedding space, not the dimensionality of the manifold. Figure 2 below provides a visual explanation.

2) CALCULUS ON MANIFOLD

In this subsection, we consider functions defined on manifolds, starting with a smooth real function defined in scalar

field $f : \mathrm{X} \to \mathbb{R}$; then we introduce a mapping function $F : \mathrm{X} \to T\mathrm{X}$ defined in the tangent vector domain that associates a tangent vector $F(x) \in T_x\mathrm{X}$ with a point x. The tangent vector field is used to formalize the concept of infinitesimal displacement on manifold. We use $L^2(\mathrm{X})$ and $L^2(T\mathrm{X})$ to represent Hilbert Spaces of scalar field and vector field on manifold respectively, and define their inner product operation as follows:

$$\langle f, g\rangle_{L^2(\mathrm{X})} = \int_{\mathrm{X}} f(x)g(x)dx \tag{13}$$

$$\langle F, G\rangle_{L^2(T\mathrm{X})} = \int_{\mathrm{X}} \langle F(x), G(x)\rangle_{T_x\mathrm{X}}dx \tag{14}$$

where dx denotes an area element (d-dimensional) induced by the Riemannian metric.

Define the differential of f as the operator acting on the tangent vector field $df : T\mathrm{X} \to \mathbb{R}$. In order to model a small displacement near the point x, at each point x, the differential operation is defined as $df(x) = \langle \nabla f(x), \cdot\rangle_{T_x\mathrm{X}}$ applied to the tangent vector $F(x) \in T_x\mathrm{X}$. So we can apply this definition of differentiation to the tangent vector, that is $df(x)F(x) = \langle \nabla f(x), F(x)\rangle_{T_x\mathrm{X}}$, and get the change in function value due to a small displacement around point x, where $\nabla f : L^2(\mathrm{X}) \to L^2(T\mathrm{X})$ is called intrinsic gradient. As we can see that it is similar to the classical notion of a gradient, because it defines the direction of the tangent vector that changes most dramatically at a local given point in a function.

After having a definition of gradients, we also need to define the divergence operation as the intrinsics divergence operation, which acts on the tangent vector field $\mathrm{div} : L^2(T\mathrm{X}) \to L^2(\mathrm{X})$, adjoint to the gradient operator:

$$\langle F, \nabla f\rangle_{L^2(T\mathrm{X})} = \langle -\operatorname{div} F, f\rangle_{L^2(\mathrm{X})} \tag{15}$$

Finally, we can obtain Laplacian (or Laplace-Beltrami operator in differential geometry) acting on scalar fields, that is, the divergence of gradient can be taken as negative. $\Delta : L^2(\mathrm{X}) \to L^2(\mathrm{X})$ is an operator defined as follows:

$$\Delta f = -\operatorname{div}(\nabla f) \tag{16}$$

combine the equations (14) with (12), it can be concluded that Laplacian is symmetric:

$$\langle \nabla f, \nabla f\rangle_{L^2(T\mathrm{X})} = \langle -\operatorname{div}(\nabla f), f\rangle_{L^2(\mathrm{X})} = \langle \Delta f, f\rangle_{L^2(\mathrm{X})} \tag{17}$$

$$= \langle f, \Delta f\rangle_{L^2(\mathrm{X})} \tag{18}$$

The summary is that we can think of Laplacian as the difference between the average of a function on an infinitesimal sphere around point x and the value of the function on x. The Laplace-Beltrami operator (LBO) is intrinsic, *i.e.* expressible entirely in terms of the Riemannian metric. As a result, it is invariant to isometric (metric-preserving) deformations of the manifold. On a compact manifold, LBO admits the same rules of spectral analysis on graphs. According to the background knowledge in the previous part, it can be seen that Laplacian plays a crucial role in signal processing and learning in the non-Euclidean domain, helping to generalize the classical Fourier transform and spectral analysis on graph and manifold data.

3) DISCRETE MANIFOLDS

In computer graphics, two-dimensional manifolds are commonly used to model 3D shapes and and it is difficult to reconstruct a correct discretization of a manifold from point cloud. The common way to discrete a manifold is to sample the manifold by N points with coordinates $\mathbf{x}_1, \cdots, \mathbf{x}_N$ and build a graph on this poind cloud in which the points are vertices and the edges with weights represent the local connectivity in the manifold. However, this simple discretization method cannot directly obtain the geometry of the bottom continuous manifold when the sampling density increases and the Laplacian operator of the graph does not converge to the continuous Laplacian operator of the manifold. Another way is that in the discrete setting, the points on surface X are constructed by a triangular mesh (V, E, F) with vertices $V = \{1, \cdots, N\}$, the triangular faces ijk and $ijh \in F$ have edges $\{(i,j), (i,k), (j,k)\} \in E$. The collection of faces represents the underlying continuous manifold as a polyhedral surface consisting of small triangles glued together. To be a correct discretization of a manifold (a manifold mesh), every edge must be shared by exactly two triangular faces. If the manifold has a boundary, any boundary edge must belong to exactly one triangle. On a triangular mesh, the simplest discretization of the Riemannian metric is given by assigning each edge a length $l_{ij} > 0$, which must additionally satisfy the triangle inequality in every triangular face. If $l_{ij} = \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2$, the mesh Lapalcian is given as the cotangent weights $w_{ij} = \frac{1}{2}\left(\cot\alpha_{ij} + \cot\beta_{ij}\right)$.

E. CONVOLUTIONAL NEURAL NETWORKS

In the Euclidean domain, the convolution operation can be considered as passing a template at each point in the domain and recording the correlation between the template and the function at that point due to shift invariance. With image filtering operation in mind, this is equivalent to taking a block of pixels (usually a square block), multiplying its elements by using a template, summing up the result, and sliding the window to the next position. Figure 3 shows the process. The shift invariance here means that the operation of extracting patch in each location is always the same. CNN techniques [32]–[34] allows to learn task-specific features from examples and achieves a breakthrough in performance in a wide range of applications such as image classification [6], [35], [36], segmentation [37], detection and localization [38], [39] and annotation [40].

III. METHODS

With the background knowledge introduced, we now come to focus on the questions of constructing CNNs on non-Euclidean domains. At present, the two typical non-Euclidean geometric data in research are graphs and manifolds. Graphs refer to network structure data composed of nodes and edges, such as social network. Manifolds are

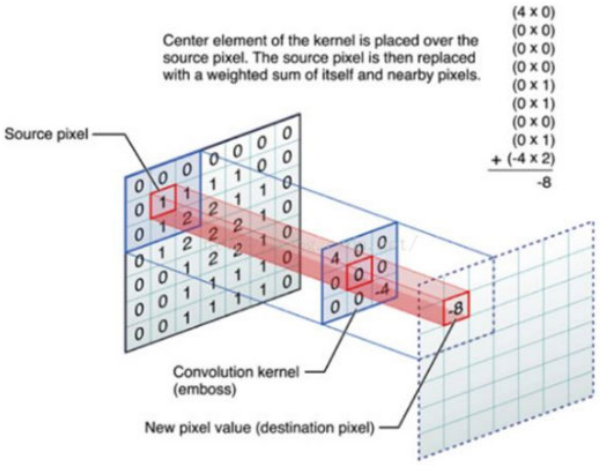

FIGURE 3. Convolutional layer in traditional CNN.

often used to describe 3D geometric shapes, such as spatial coordinates on the surface of an object returned by a LiDAR scan. In this section, we will introduce various methods and network architectures used to deal with graphs and manifolds.

A. METHODS ON GRAPHS

Graph has been a common structure of data in the real world. There are lots of tasks that can be described as problems on the graphs, such as social network, protein structure network, traffic network, and knowledge graph, and even some regular data like grid structural data (such as images, video, etc.) are also a special form of graph data. In recent years, driven by the development of social networks and knowledge graphs and inspired by the great success of deep learning techniques in computer vision field, many researchers have imitated and designed the deep neural network architectures in graph data by referring to the original Convolutional Neural Network (CNN), Long Short-Term Memory network (LSTM), Attention mechanism and Auto-encoder (AE).

However, it is important to note that graph data are different from images due to its irregularity. Each node in the graph may have different number of neighbors. Convolution operations on images may not be applied to graph data. On the other hand, the nodes in a graph are related to each other which may conflict the data independence assumption in common machine learning algorithms. Another aspect is the data size. A graph may have a large number of nodes, which pose a huge challenge to existing machine algorithms.

In this section, we will focus on the major spectral-based and spatial-based graph neural networks (GNN) that have been developed recently and extend to Graph Attention Networks (GANs), Graph Generative Networks (GGNs) and Graph Auto-encoders (GAEs).

The researchers aim to generalize the traditional convolution operation from images or grids to the graphs, the key of this generalization process is to learn a function f which has ability to aggregate the node v_i's features $\mathbf{X}_i$ and features from neighbors v_j's features $\mathbf{X}_j$ to create a new representation of node v_i. Generally speaking, the graph convolutional networks can be divided into two categories. One is graph convolution based on spectral graph theory which is inspired by graph signal processing techniques [31]. The researchers define graph convolution operation by introducing Fourier transform on graph, which can also be interpreted as an operation to remove noise from graph signal, then combine with deep learning techniques. The other one is spatial domain convolution, which directly carries out convolution operation on the graphs. This kind of GCN approach represents graph convolution as the combination of feature information from neighbor nodes.

1) SPECTRAL-BASED GRAPH CONVOLUTIONAL NETWORKS

The initial core idea of this kind of methods is to define non-Euclidean convolution and its relation with frequency domain by analogy. This method first uses Fourier transform to multiply in frequency domain, then do inverse transformation. After a few years of development, Fourier transform is not needed any more, graph Laplacian is applied directly. The major differences between the different frameworks lie in the design of filters in convolutional layer.

Spectral CNN (SCNN) [12]: Inspired by the definition of the Fourier transform on graph, J. bruna *et al*. proposed the first graph convolutional neural network (Spectral CNN). For irregular graph data, we can use the Fourier transform formula of the graph to take the product of the input in Fourier domain first, and then take the inverse Fourier transform, which is equivalent to directly applying the convolution in the original space. In [12], the filters are defined as:

$$\mathbf{g}_\theta = \boldsymbol{\theta} = \begin{pmatrix} \theta_1 & & \\ & \ddots & \\ & & \theta_N \end{pmatrix} \tag{19}$$

where $\mathbf{g}_\theta$ is an $N \times N$ diagonal matrix of spectral multipliers representing a learnable filter in the frequency domain, and the parameter $\boldsymbol{\theta} \in \mathbb{R}^N$ is a vector of Fourier coefficients. A signal $\mathbf{x}$ filtered by $\mathbf{g}_\theta$ can be written as:

$$\begin{aligned} \mathbf{y}_{output} &= \mathbf{x}_G^* \mathbf{g}_\theta = \sigma\left(\mathbf{U}\left(\mathbf{g}_\theta\left(\mathbf{U}^{\mathrm{T}}\mathbf{x}\right)\right)\right) \\ &= \sigma\left(\mathbf{U}\begin{pmatrix} \theta_1 & & \\ & \ddots & \\ & & \theta_N \end{pmatrix}\mathbf{U}^{\mathrm{T}}\mathbf{x}\right) \end{aligned} \tag{20}$$

where $\mathbf{U}$ is an $N \times N$ matrix of Laplacian eigenvectors, and σ is a non-linearity function (*e.g.* ReLU) applied on the vertex-wise function values. The pooling operation in traditional CNN framework is replace by graph coarsening procedure in this SCNN model, that is, input a graph with N vertices, and then produces a graph with $N' < N$ vertices and transfers signals from the vertices of the fine graph to those of the coarse one. The coarsening operation can also be regarded as the resolution change. The commonly used coarsening algorithm is Graclus.

Despite SCNNs has groundbreaking theoretical contributions, the network has several major drawbacks. First, the computation of the forward and inverse graph Fourier transform incurs intensive multiplications. Second, the spectral filter coefficients are based on the basis and the Laplacian of each graph is different, therefore, the model of the spectral CNN cannot achieve good portability. Third, the model does not have a good spatial localization. Unlike traditional convolution neural network where the convolutional operation only sees a fraction of the field each time, spectral CNN considers every single node with each convolution process. Fourth, because the non-linear functions in (20) are applied to the spatial domain, graph Laplacian eigenvectors at each resolution must be recalculated in practice and applied directly after each pooling step, which caused a lot of trouble with the calculation.

Smooth Spectral CNN (Smooth SCNN) [13]: In later work, Henaff *et al*. found that it is necessary to restrict the class of spectral multipliers to those corresponding to localized filters. They expressed the spatial localization of filters in the frequency domain and considered that smooth spectral filter coefficients can constitute a spatial localized filters. Using only the first K eigenvectors in (20) sets a cutoff frequency. In general, $K \ll N$, since only the first K Laplacian eigenvectors describing the smooth structure of the graph are useful in practice. This reduces the number of parameters to learn. Thus, the parametric filters are defined as the following form:

$$\mathbf{g}_\theta = \mathbf{B}\boldsymbol{\theta} \tag{21}$$

where $\mathbf{B} \in \mathbb{R}^{N\times K}$ is a fixed interpolation kernel (*e.g.* $\mathbf{B}$ can be the cubic B-spline basis) and the parameter $\boldsymbol{\theta} \in \mathbb{R}^K$ is a vector of interpolation coefficients.

Chebyshev Spectral CNN (ChebNet) [15]: Mathematically, it is known that any polynomial of order k can be expanded by using the Chebyshev polynomial. Defferrard *et al*. proposed ChebNet framwork by using the Chebyshev polynomial basis to represent the spectral CNN's filters. The Chebyshev polynomial in recursive form can be written as:

$$\begin{aligned} T_0(\lambda) &= 1, \\ T_1(\lambda) &= \lambda, \\ &\vdots \\ T_k(\lambda) &= 2\lambda T_{k\text{-}1}(\lambda) - T_{k\text{-}2}(\lambda). \end{aligned} \tag{22}$$

Therefore, after combining the recursive form of Chebyshev polynomial, the filters can be parametrized as the truncated expansion of order $K-1$:

$$\mathbf{g}_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}) \tag{23}$$

where the parameter $\boldsymbol{\theta} \in \mathbb{R}^K$ is a vector of Chebyshev coefficients and Λ is a diagonal matrix composed of Laplacian eigenvalues. $T_k(\tilde{\Lambda}) \in \mathbb{R}^{N\times N}$ is the Chebyshev polynomial of order k evaluated at $\tilde{\Lambda} = 2\Lambda/\lambda_{\max} - \mathbf{I}_N$, aiming to reduce the size of the eigenvalues from $[0, \lambda_N]$ to $[-1, 1]$, since the Chebyshev polynomial form an orthonormal basis that lies in $[-1, 1]$. Therefore, the ChebNet's filtering operation can be written as:

$$\begin{aligned} \mathbf{y}_{output} &= \mathbf{x}_G^* \mathbf{g}_\theta = \sigma\left(\mathbf{U}\left(\mathbf{g}_\theta\left(\mathbf{U}^{\mathrm{T}}\mathbf{x}\right)\right)\right) \\ &= \sigma\left(\sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{L}})\mathbf{x}\right) \end{aligned} \tag{24}$$

where σ is a non-linear function and $T_k(\tilde{\mathbf{L}}) \in \mathbb{R}^{N\times N}$ is the Chebyshev polynomial of order k evaluated at the scaled Laplacian $\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{\max} - \mathbf{I}_N$. In addition, the property $\mathbf{U}\Lambda^k\mathbf{U}^{\mathrm{T}} = \mathbf{L}^k$ of Laplacian matrix eigen-decomposition is also used in formula (24). In short, the essence of ChebNet is that every graph newly generated by coarsening algorithm (Graclus, built on Metis) has its own Laplacian matrix $\mathbf{L}$. All that the model needs to do is to select several $\mathbf{L}$ from them and generate the convolutional layer filter to process the graph.

We can clearly see that θ_k is the learnable parameter and consistent with $T_k(\tilde{\mathbf{L}})$, which corresponds to the order k neighbor. This means that parameters are shared on neighborhood of the same order, but not shared on neighborhood of different orders. As the result of $T_k(\tilde{\mathbf{L}})$ is $\mathbf{L}$'s polynomial of order k, which only considers the *K-hop* neighborhood. Therefore, $T_k(\tilde{\mathbf{L}})\mathbf{x}$ can operate locally on each node, which means that the filters of ChebNet is localized in spatial domain. However, because there are only K parameters, it is difficult for the model to allocate different weights for different nodes in the same order neighborhood.

Graph Convolutional Networks (GCNs) [16]: GCN simplifies ChebNets architecture by using filters operating on *1-hop* neighborhoods of the graph. Kipf and Welling *et al*. introduced one simplified version, namely, first-order approximation of ChebNet [15]. The innovations can be summarized in two points:

- let K=1, $\lambda_{\max}$=2, that is, each layer of convolution only considers the direct neighborhood, which is similar to the 3×3 kernel in CNN
- deepen the network and reduce the width of the model, which is widely used in conventional deep learning for images.

Under this condition, the output of convolution operation is:

$$\begin{aligned} \mathbf{y}_{ouput} &= \mathbf{x}_G^* \mathbf{g}_\theta = \sigma\left(\theta_0\mathbf{x} + \theta_1\left(\mathbf{L} - \mathbf{I}_N\right)\mathbf{x}\right) \\ &= \sigma\left(\theta_0\mathbf{x} - \theta_1\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right) \end{aligned} \tag{25}$$

where the derivation process of formula applies the normalized Laplacian $\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-\frac{1}{2}} = \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$. In order to prevent the occurrence of overfitting due to the excessive number of parameters, a unified hypothesis is used $\theta = \theta_0 = \theta_1$. Then the definition of GCN becomes:

$$\mathbf{y}_{output} = \mathbf{x}_G^* \mathbf{g}_\theta = \sigma\left(\theta\left(\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathbf{x}\right) \tag{26}$$

In order to fuse multi-dimensional input graph signals and satisfy multi-channel convolution, the graph convolutional layer is proposed by modifying equation (26):

$$\mathbf{X}^{k+1} = \tilde{\mathbf{A}}\mathbf{X}^{k}\mathbf{\Theta} \tag{27}$$

where $\tilde{\mathbf{A}} = \mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$; $\mathbf{X} \in \mathbb{R}^{N\times C}$, $\mathbf{\Theta} \in \mathbb{R}^{C\times F}$, $\mathbf{X}^{k+1} \in \mathbb{R}^{N\times F}$, among which N, C and F represent the number of nodes, channels and convolution kernel respectively.

CGN is spatially localized, bridging the gap between spectral-based and spatial-based methods. Each row of output matrix is calculated by weighted aggregation of a node itself and its neighboring nodes, which represent the hidden representation of a node. However, the disadvantages of this model lie in: 1) the weights assigned to different neighbors in the same order neighborhood are exactly the same, which limits the ability of the model to capture the correlation of spatial information. This problem was solved by Zhang *et al.* 's work [41], which greatly improved the characterization ability of spatial correlation in the graph datasets with a small scale. So arbitrary weight coefficients can be assigned to arbitrary neighbors. 2) during batch training of GCN, the calculation consumption increases exponentially as the number of model layers increases.

Adaptive Graph Convolution Networks (AGCNs) [42], [43]: Li *et al.* proposed graph adaptive convolutional network (AGCN), which applies a so-called residual graph in graph augmentation process in which residual graph is constructed by computing the pairwise distance between nodes. AGCN can capture complement relational information and hidden structural relations unspecified by the graph Laplacian matrix.

GCN with Complex Rational Spectral Filters (CayleyNets) [17]: The core component of this proposed network model is the new Cayley polynomial (parametric rational complex function). Based on this, an efficient spectral filtering scheme is constructed, which has the advantages of edge positioning and linear complexity similar to Chebyshev filter. But one of the main disadvantages of the Chebyshev filter is that it is difficult to generate some desired narrow-band filters because such filters require a high order K to support. Therefore, the main advantage of Cayley filters over Chebyshev filters is that they can detect important narrow bands in the training process, and can locate these bands well on the graph, allowing effective calculation for the interested bands. The Cayley filters are positioned in space in a linear relation to the size of the input data for the sparsely connected graph and are capable of handling different structures of the Laplacian operator. Experiments show that CayleyNets have greater flexibility and perform better than ChebNets on a wide range of graph learning problems.

The Cayley polynomial of order K is defined as a real-valued function form as follows:

$$T_{c,h}(\lambda) = c_0 + 2\,\mathrm{Re}\left\{\sum_{k=0}^{K-1} c_k(h\lambda - i)^k(h\lambda + i)^{-k}\right\} \tag{28}$$

where $\mathbf{c} = (c_0, \cdots, c_{K-1})$ is a vector of real coefficients, and $h > 0$ is the spectral zoom parameter. By adjusting the spectral zoom parameter h, different parts of the spectrum can be "zoomed" to produce filters specifically for different frequency bands. Therefore, the Cayley filter $\mathbf{g}_\theta$ is a spectral filter defined on real signals $\mathbf{f}$ by:

$$\begin{aligned}\mathbf{g}_\theta\mathbf{f} &= T_{c,h}(\Delta)\mathbf{f} \\ &= c_0\mathbf{f} + 2\,\mathrm{Re}\left\{\sum_{k=0}^{K-1} c_k(h\Delta - i\mathbf{I})^k(h\Delta + i\mathbf{I})^{-k}\mathbf{f}\right\}\end{aligned} \tag{29}$$

where the parameters $\mathbf{c}$ and h are optimized during training. It is also observed that the filtering operation $\mathbf{g}_\theta\mathbf{f}$ eliminates the actor eigen-decompostion process of the Laplacian operator, which significantly reduces the operational costs. The Cayley filters are special cases of ARMA filters which are based on general rational functions of the Laplacian.

In general, the CayleyNet architecture is a new class of complex rational Cayley filters that are localized in space, can represent any smooth spectral transfer function, and are highly regular.

2) SPATIAL-BASED GRAPH CONVOLUTIONAL NETWORKS

Spatial convolution is the direct execution of convolution operations on graph. As the size of traditional convolution kernel is fixed, if we apply traditional convolution on graph, we need select a neighborhood of a fixed size for convolution. However, unlike with data of regular grid structure, the nodes in the graph usually have different number of neighborhoods. Convolution operation on graph imitates the convolution operations on images and is defined based on spatial relations of nodes. Similar to the center pixel in the traditional CNN 3×3 filters, the representation of a center node is also based on the aggregation result of its neighbor node.

Graph Neural Networks (GNNs) [11]: The basic idea of GNN is recursively updating the hidden representation of nodes until convergence. Therefore, GNN is also referred to as recurrent-based spatial GCN, which uses the same graph convolution layer to update the hidden representation. From diffusion process perspective, each node exchanges information with its neighbors until reaching the equilibrium state. To process the heterogeneous graph, the spatial graph convolution of such GNN is defined as:

$$\mathbf{h}_v^k = f\left(\mathbf{X}_v, \mathbf{X}_{co}[v], \mathbf{h}_{N(v)}^{k-1}, \mathbf{X}_{N(v)}[v]\right) \tag{30}$$

where $\mathbf{X}_v$ denotes the label attribute of node v, $\mathbf{X}_{co}[v]$ represents the label attribute of the edge corresponding to node v, $\mathbf{X}_{N(v)}[v]$ represents the label attribute of the neighbor node of node v, $\mathbf{h}_{N(v)}^{k-1}$ represents the hidden representation of the neighbor nodes of node v at time step k, $f(\cdot)$ is a recursive function, and the goal is to make sure that the final result converges. GNN adopts almeida-pineda algorithm [44], [45] to train its model and its key idea is to run the forward propagation and back propagation until convergence.

GraphSage [46]: GraphSage introduces aggregation function to define the convolution in the spatial domain on graph

and learns the embedding of each node in an inductive way. The aggregation function is essentially the aggregation of the neighborhood information of the nodes, and its output should be invariant to the order of nodes. That is, the input order will not affect the output result of the aggregation function, such as mean, sum and max function. In this way, each node is represented by the aggregate result of its neighborhood. Different graph convolution layers are used to update the hidden representation of these spatial-based GCNs. The graph convolution operation is defined as:

$$\begin{aligned} \mathbf{h}_{N(v)}^{k} &= \text{AGGREGATE}_k\left(\left\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\right\}\right) \\ \mathbf{h}_v^{k} &= \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}\left(\mathbf{h}_v^{k-1}, \mathbf{h}_{N(v)}^{k}\right)\right) \end{aligned} \tag{31}$$

where $k \in [1, K]$ is the number of iterations, and all nodes $v \in \mathbf{V}$ are updated during each iteration by formula (31), σ is a non-linear activation function, AGGREGATE_k represents the differentiable aggregator function, $N(v)$ is the neighboring nodes of node v, $\mathbf{h}_v^k$ denotes the hidden vector representation of node v when the model updating iteration comes to k times, $\mathbf{W}^k$ is a weight matrix. Instead of updating status on all nodes at a time, GraphSage proposes a batch learning algorithm that improves scalability for large scale graphs. Basically, GraphSage's learning process consists of three steps:

1) sample the local *k-hop* neighborhood of nodes at a fixed size.
2) obtain the final state of the center node by aggregating the neighbor feature information of the center node.
3) use the final state of the center node for prediction and back propagation of errors.

Diffusion CNN (DCNN) [14]: Atwood and Towsley proposed a spatial-based graph CNN architecture applying a diffusion (random walk) process on the graph. The transition probability of a random walk on a graph is given by $\mathbf{P}_t = \mathbf{D}_t^{-1}\mathbf{A}_t$, where $\mathbf{A}$, $\mathbf{D}$ are adjacency matrix and degree matrix respectively and t is the time index.

DCNN is mainly used in node classification or graph classification. For each task, the object (node or graph) is transformed to a diffusion-convolutional representation $\mathbf{Z}_t$, which is a $K \times F$ real matrix defined by K hops of graph diffusion over F features.

For the node classification task, the convolution formula is defined as:

$$\mathbf{Z}_{tikf} = \sigma\left(\mathbf{W}_{kf} \cdot \sum_{j=1}^{N_t} \mathbf{P}_{tikj}^{*}\mathbf{X}_{tjf}\right) \tag{32}$$

where the t th graph is described by an $N_t \times F$ feature matrix $\mathbf{X}_t$, $\mathbf{X}_{tjf}$ represents the f th feature of j th node, $f \in [1, F]$, N_t denotes the number of nodes of G_t, $\mathbf{P}_t^*$ is an $N_t \times K \times N_t$ tensor, $\mathbf{P}_{tikj}^{*} = \mathbf{P}_{tif}^{k}$ denotes the probability of G_t's node v_i reaching node v_j after k hops, $\mathbf{P}_t^k$ denotes the k hops transfer matrix for graph G_t, $k \in [1, K]$, $\mathbf{W}$ is a $K \times F$ real-valued weight matrix, which is the learnable parameters of convolution kernels, σ is a non-linear differentiable activation function. Here, the diffusion-convolutional representation of G_t, $\mathbf{Z}_t$ is a $N_t \times K \times F$ tensor. So the formula (32) is expressed as a tensor in the form of $\mathbf{Z}_t = \sigma\left(\mathbf{W} \odot (\mathbf{P}_t^* \mathbf{X}_t)\right)$.

For the task of graph classification, the convolution formula is defined as:

$$\mathbf{Z}_t = \sigma\left(\mathbf{W} \odot 1_{N_t}^{\mathrm{T}} \mathbf{P}_t^* \mathbf{X}_t / N_t\right) \tag{33}$$

where 1_{N_t} is an $N_t \times 1$ vector of ones, and the diffusion-convolutional representation of G_t here, $\mathbf{Z}_t$, is a $K \times F$ matrix. DCNNs are learned via stochastic minibatch gradient descent on back-propagated error.

PATCHY-SAN [9]: In this method, the graph structural data are transformed into grid structural data through graph labeling procedure, and uses traditional CNN to solve graph classification tasks, so as to maintain shift-invariance properties. The graph labeling procedure is essentially to sort each node in graph according to the node's degree, centrality and etc. The core idea of PATCHY-SAN can be divided into three steps:

1) determine fixed number of nodes for each graph by using graph labeling procedure.
2) choose and sort a fixed number of neighbor nodes for each node according to the results in the first step, so as to generate grid-structured data of fixed size.
3) use the traditional CNN to learn graph's hidden representation. However, the shift-invariance nature depends on the sorting function, and underlying graph labeling procedure only considers the structure of graph and ignores feature information of nodes.

Large-scale Graph Convolution Networks (LGCN) [47]: To solve the problem faced by PATCHY-SAN, LGCN proposed a sorting method based on node feature information. PATCHY-SAN has a complex preprocess step in graph labeling procedure and the graph structures only consider shallow layers, With a sorting method based on node feature information proposed, LGCN is more efficient and can provide deep expression. In addition, LGCN also proposed a subgraph training strategy to adapt the network model to handle large-scale graph data, which is to take the sampled small subgraph as mini-batch for training.

LGCN first uses traditional CNN to generate the node-level output. Secondly, for each node, LGCN assembles a feature matrix of its neighbor node, and sorts the matrix along each column. The first k rows of the matrix serve as the input grid data of the target node. Finally, LGCN uses 1D CNN to process the output of the previous step to obtain the hidden representation of the target node.

Mixture Model Networks (MoNet) [23]: In this work, the authors proposed a unified framework that can extend the CNN structure to non-Euclidean domains (graphs and manifolds), and can learn local, stationary and combinatorial specific task characteristics. Meanwhile, it also claimed that various non-Euclidean CNN methods proposed in previous literatures can be regarded as a special case of MoNet. See MoNet in detail in subsection III-A.2.

3) COMPARISON BETWEEN THE SPECTRAL AND SPATIAL CONVOLUTION APPROACHES

The differences between the spectral and spatial convolutional approaches will be introduced next.

Firstly, the efficiency of spectral-based models is higher than that of spatial-based models. The computation cost of spectral-based models changes with the size of graphs, which makes it difficult for them to parallelize or scale to large graphs. Moreover, the operation of eigen-decomposition is required for the spectrorization-based method, which makes the computational cost and time cost increase greatly. However, spatial-based approaches enable the models to handle large-scale graph and achieve the localization by clustering neighboring nodes and performing convolution operations directly on graph.

Secondly, spectral-based model usually focuses on a fixed graph, which makes it difficult to extend the application to other graphs, because the nature of each graph and Laplacian is unique. But the spatial-based model partially performs graph convolution on each node, so parameter sharing can easily be realized between different positions and structures. This makes the spatial-based model more general than the spectral-based model.

Finally, it is the biggest problem with the spectral-based model that the model is only able to deal with undirected graph, and the only way to apply the model to the directed graph is to convert it to undirected graph, and the spatial based model can deal with multi-source inputs, such as edge features and edge directions, in a more flexible manner. Therefore, in recent years, spatial-based models are still the main research direction and have received more and more attention.

4) BEYOND GRAPH CONVOLUTIONAL NETWORK ARCHITECTURES

In this part, we will summarize other graph neural networks, including graph attention neural networks, graph generative networks and graph auto-encoders.

Graph Attention Networks (GANs): The attention mechanism has been widely used in sequential tasks [48]. In many applications, such as machine translation and natural language understanding, the effectiveness of attention mechanism has been proved. Its advantage is that it can focus on the most important part of the target. Graph neural networks can also use attention mechanism in the aggregation process, integrate the output of multiple models, and generate importance-oriented random walk. The graph attention networks (GATs) proposed in [49] is a spatial-based graph convolution network in which attention mechanism is used to determine the importance of each neighbor node to the center node when the neighbor information of nodes is aggregated, *i.e*. weights. The work of [50] proposed a recursive neural network model in order to solve the graph classification problem, and the information-rich part of graph is processed by adaptive access to important node sequences.

Graph Generative Networks (GGNs): The graph generative networks (GGNs) generate a new graph with a given set of observed under the premise of the graph. At present, most methods of graph generation are directly related to a specific field, for example, in natural language processing, semantic graph or knowledge graph is generated under the condition of a given sentence [51], [52]. About generating method, it is easy to be associated with the ideas of generative adversarial networks such as NetGAN model combined with LSTM and Wasserstein GAN, which uses the generator and discriminator to generate or differentiate fake random walks as best as possible [53]. The co-occurrence matrix of the node in the random walks collection is trained to achieve normalization, so as to obtain the new graph. And some of the other methods regard the entire generation process as alternating nodes and edges, such as [54]. GraphRNN model proposed in this work uses two recurrent neural networks, *i.e*. graph-level and edge-level. The former network adds a new node to the sequence of nodes each time, and the latter network generates a binary sequence that represents the connection between the newly added node and the previously generated node in the sequence.

Graph Auto-encoder (GAEs): Graph auto-encoders is also called graph embedding or graph representation learning. A typical approach is to use multilayer perceptron as an encoder to obtain embedding results of nodes, and then the decoder restructures the neighborhood statistics of nodes accordingly. The goal is to use neural network architecture to express the network vertices transformation in low dimensional vector space.

Recently, Kipf *et al*. integrated GCN models [16] into graph auto-encoder framework and defined the encoder as $\mathbf{Z} = GCN(\mathbf{X}, \mathbf{A})$ and the decoder as $\hat{\mathbf{X}} = \sigma\left(\mathbf{Z}\mathbf{Z}^{\mathrm{T}}\right)$ [55]. The adversarially regularized graph auto-encoders (ARGAs) proposed in [56] can be thought of as a combination of GCN and GAN where the generative adversarial network training scheme was used to train the generator and discriminator. In this work, the encoder uses the features of the graph nodes to encode its structural information into the hidden representation of GCN, and the decoder reconstructs the adjacency matrix from the encoder's output. In the AGRA model, the encoder can be regarded as a generator to generate the "fake samples" as real as possible, the decoder can be regarded as a discriminator to identify "fake samples" from the real samples as much as possible, that is, to identify the hidden representation of all nodes as much as possible.

In addition, the structural deep network embedding (SDNE) [57] uses the stacked auto-encoders approach, while preserving first-order and second-order proximity of nodes. The deep recursive network embedding (DRNE) [58] reconstructs the hidden state of nodes without choosing to reconstruct the entire graph's statistical information, and LSTM is used as encoder as aggregation function where the neighbors' sequence is ordered by their node degree.

FIGURE 4. Illustration of the difference between extrinsic (left) and intrinsic (right) deep learning methods on geometric data. Intrinsic methods work on the manifold rather than its Euclidean realization and are isometry-invariant by construction.

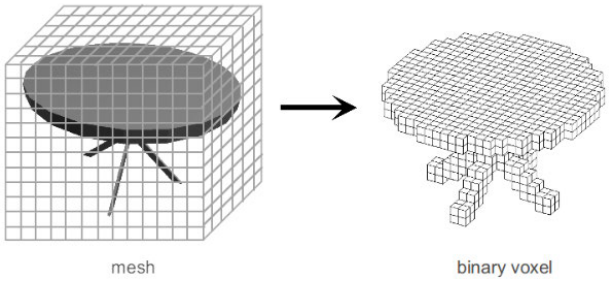

FIGURE 5. Architecture of 3D ShapeNets model. Cited from [62].

B. METHODS ON MANIFOLDS

Manifold is often used to describe 3D shapes, in which the shape descriptors play an important role. Generally speaking, a local feature descriptor assigns a vector to each point on the shape in a multi-dimensional descriptor space, representing the local structure of the shape around that point [59], [60] and a global feature descriptor is constructed by aggregating local descriptors to describe the whole shape's geometric properties, *e.g.* using the bag-of-features paradigm [61]. Descriptor construction depends on specific application tasks. There are two fundamental questions needed to be considered when we construct or choose a descriptor: which shape features do the descriptors have to capture; and to which transformations of the shape shall it remain invariant or insensitive.

In past ten years, we have witnessed the emergence of learning-based approaches in 3D shape analysis tasks. All this progress, of course, is due to the emergence of convolutional neural network (CNN) techniques which allow to learn task-specific features [6]. Currently, almost all the methods in deep learning field dealing with 3D shape are basically applied under CNN paradigm. Methods like [62], [63] directly use the standard (Euclidean) CNN architectures in neural networks which is applied to volumetric 2D multi-view shape representations. The shape descriptors in these methods are based on extrinsic structures that are invariant under Euclidean transformations, making them unsuitable for dealing with deformable shapes. While other methods [18], [19], [21], [23], [64] design a new framework by imitating the CNN feature extraction pattern to explore the intrinsic versions of CNNs that would allow dealing with shape deformations. However it is difficult to formulate due to the lack of shift invariance on Riemannian manifolds. These methods can be classified into two categories, *i.e.* extrinsic and intrinsic.

Extrinsic Deep Learning:

Many previous machine learning techniques have tried to use a standard framework to process images as if they were actually being processed directly on 3D data. However, the biggest disadvantage of these methods is that they still treat geometric data as Euclidean data. The extrinsic data representation does not perform well with the change of the object position or shape. Moreover, such methods need complex models and a lot of training to support the task of realizing the invariance of shape deformation, which is very difficult in practice.

1) VOXEL-BASED REPRESENTATION METHOD

Voxel data is one of the data representation of 3D data, in which a voxel is a smallest data unit in 3D space. Therefor, the entire 3D object can be divided into a 3D grid. Similar to 2D image, voxelization is carried out at a certain a resolution, *i.e.* the size of each grid cell. The finer the 3D space is divided, the smaller each grid is, and the greater the resolution is.

3D ShapeNets (3DSN) [62]: This work proposed a typical network architecture to process voxel data for 3D objects. Wu *et al*. applied CAD data as training data, constructed 3D ShapeNets by designing Convolutional Deep Belief Network (CDBN). The 3D ShapeNets learned the joint distribution of input voxel data x and object category label y, and identified the target and restored the full 3D shape using the 2.5D depth map obtained by Kinect sensors. The core of this method is stereoscopic voxelizing based on 3D data, and the 3D data of object is represented as a voxel with the size of $30 \times 30 \times 30$. In addition, the author also designed a next-best-view-prediction system to allow observers to observe objects from another perspective. When the recognition fails, we can greatly reduce the uncertainty of recognition from the first perspective. Given the current view, the model is able to predict which next view would be optimal for discriminating the object category. In order to reduce the effect on the shape, the author does not add pooling operation layer. Figure 5 shows the architecture composition and usage instructions of the 3D ShapeNets model, respectively.

The problem of such an approach is the large memory consumption and long computation time during training. In future work, there are three directions to improve volumetric CNNs:

1) Design of new network structures. The first network reduces overfitting by introducing auxiliary learning tasks, which help to scrutinize details of 3D objects more deeply by using partial subvolumes to predict the categories of objects. The second network imitates multi-view CNNs. As they are strong in 3D shape classification tasks, but the data are not "shot" through by a different path and multiple angle of view 2D rendering,

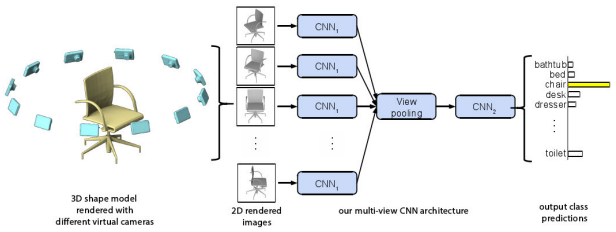

FIGURE 6. Procedure of Multi-view CNN model. Cited from [63].

but the 3D shape data through a long and anisotropic probing convolution kernel. 3D data are projected onto 2D data, and then 2D images are processed by 2D CNNs to achieve the image classification task.

2) Data augmentation. By augmenting the training data with different azimuth and elevation rotations to improve model performance.
3) Feature pooling. The two new network structures mentioned above all capture different kinds of information in different directions, so we can add a orientation pooling stage that aggregates information from marketers orientations.

2) MULTIVIEW-BASED CNNS METHOD

This is different from the method that converts 3D shape representation to voxel grid representation. The central idea of multi-view neural network in the classification and segmentation of 3D shapes is to extract the surface features of 3D shapes with multiple 2D images from different angles. Then, 2D CNN technology is used to directly process the corresponding 2D images to identify and detect 3D objects. In this way, the mature CNN technology on 2D images can be directly used.

Multi-view CNN (MVCNN) [63]: Similar to the human eye to recognize 3D objects, if you cannot distinguish them from one angle, you can try to distinguish them from different angles. Multi-perspective analytical 3D model technology also takes advantage of this. The multi-view CNN architecture learns to predict 3D shapes from views of the shapes using image-based CNNs but in the context of other views via a view-pooling layer. As a result, information from multiple views is effectively accumulated into a single, compact shape descriptor. This multi-view representation is constructed in three steps:

1) a 3D shape is rendered into multiple images using varying camera extrinsics;
2) image features are extracted for each view using feature extractors like VGG or AlexNet)
3) the features are combined across views through a pooling layer, followed by fully connected layers.

The whole process of MVCNN is shown in the figure 6 below.

MVCNN can be effectively used in 3D model reconstruction, 3D object classification as well 3D object retrieval. However, the disadvantage of this method is that the object itself is blocked and the light intensity and pitch angle are different. Some surface information will be lost when taking different views, and the angle selection usually has artificial marks.

3) COMPARISON BETWEEN THE VOLUMETRIC CNN AND MULTI-VIEW CNN

The volumetric representation encodes a 3D shape as a 3D tensor of binary or real values. The multi-view representation encodes a 3D shape as a collection of renderings from multiple viewpoints. Intuitively, it seems that volumetric representation should be able to encode more feature information of 3D shapes than multi-view representation. However, the author in [65] reproduced the experiments of 3D ShapeNets (using a $30 \times 30 \times 30$ voxel grid) and multi-view CNNs (down-sampling each rendered view to 227×227) on ModelNet40 dataset [66].

Based on the results, classification accuracy under the standard framework of the two methods shows that the performance of volumetric CNN based on voxel occupancy is 7.3% worse than that of multi-view CNN. There are at least two possible reasons for this, namely, the resolution of the input data and the network architecture difference [65]. However, after data containing similar level of detail is input into the two networks, the classification accuracy of multi-view CNNs is much higher (89.5%) than that of 3D ShapeNets (84.7%). Notice that in this experiment, data of multi-view CNNs are used with renderings of the $30 \times 30 \times 30$ occupancy grid using sphere rendering, *i.e.* for each occupied voxel, a ball is placed at its center, with radius equal to the edge length of a voxel. Even with reduced details (resolution) of input data, the multi-view CNN is still much higher than 3D ShapeNets in classification accuracy. This indicates that the volumetric CNNs architecture has a lot of room to improve.

Intrinsic Deep Learning:

Different from the extrinsic methods, the intrinsic deep learning aims to promote the tradition operations to the non-Euclidean data, and use learning methods to deal with the geometric data. In the intrinsic representation, the filters are applied to 3D shape surface itself. So it will not affected by structural deformation.

4) SPATIAL-BASED METHOD

As we have seen, the inherent disadvantage of defining convolution in the spectral domain is that it does not fit across different domains. As a result, we need another kind of generalization in the spatial domain using convolution method to overcome this drawback. Traditional CNNs are able to exploit the shift-invariance and local connectivity of image data. However, to apply the CNN paradigm to the non-Euclidean domain, the lack of shift invariance is the main problem, which means the patch operator becomes position dependent.

Geodesic CNN (GCNN) [18]: Masci *et al.* proposed the first intrinsic version of convolutional neural networks on manifolds, and this approach was introduced as a generalization of CNN to triangular meshes based on geodesic local patches represented in geodesic polar coordinates. GCNN can

be interpreted as a generalization of previous popular descriptors, such as HKS [67], WKS [68], optimal spectral descriptors [59], and intrinsic shape context [69].

GCNN constructs the patch operator as:

$$(D(x)f)(\rho,\theta)=\int_{\mathrm{X}} v_{\rho,\theta}\left(x,x'\right)f\left(x'\right)dx' \tag{34}$$

The patch operator can map the value of function f near point $x\in\mathbf{X}$ to the local polar coordinates (ρ,θ), where $v_\rho\left(x,x'\right)$ represents the radial interpolation weights which is a Gaussian of the geodesic distance from x, centered around ρ, $v_\theta\left(x,x'\right)$ represents the angular weights which is constructed as a set of geodesics emanating from x in direction θ, dx' represents the area element caused by Riemannian metric, $v_{\rho,\theta}\left(x,x'\right)$ is a weighting function localized around ρ,θ.

$D(x)f$ can be regarded as a ''patch'' on the manifold and use it to define the key work, named geodesic convolution (GC):

$$(f*g)(x)=\max_{\Delta\theta\in[0,2\pi)}\int_0^{2\pi}\int_0^{\rho_{\max}} g(\rho,\theta+\Delta\theta)(D(x)f)(\rho,\theta)d\rho d\theta \tag{35}$$

where $g(\rho,\theta+\Delta\theta)$ is the filter acting on patch, the geodesic convolution is used to define an analogy of a classical convolutional layer in GCNN. Although GCNN has produced impressive results on some of the shape matching and retrieval benchmarks, but it also has some obvious drawbacks. First of all, the charting method relies on a fast marching-like procedure requiring a triangular mesh, secondly, there is no guarantee that charts will always be topological.

Anisotropic CNN (ACNN) [19]: Boscaini *et al.* used anisotropic heat kernels as spatial weighting functions allowing to extract a local intrinsic repersentation of a function defined on the manifold which is an alternative way of extracting intrinsic patches. They presented Anisotropic Convolutional Neural Networks (ACNN) is a generic convolutional neural network architecture that can be used to handle different tasks. The construction of the ''patch operator'' is much simpler, does not depend on the injectivity radius of the manifold, and is not limited to triangular meshes.

ACNN interprets heat kernels as local weighting functions and constructs the patch operator as:

$$\left(D_\alpha(x)f\right)(\theta,t)=\frac{\int_{\mathrm{X}} h_{\alpha\theta_t}\left(x,x'\right)f\left(x'\right)dx'}{\int_{\mathrm{X}} h_{\alpha\theta_t}\left(x,x'\right)dx'} \tag{36}$$

for some anisotropy level $\alpha>1$ (here $\alpha>0$ is a parameter controlling the degree of anisotropy and the situation of $\alpha=1$ corresponds to the claasical isotropic case). The values of f around point x are mapped to a local system of coordinates (θ,t) that behaves like a polar system (here t denotes the scale of the heat kernel and θ is its orientation), and $h_{\alpha\theta_t}\left(x,x'\right)$ is the anisotropic heat kernel which represents the amount of heat that is transferred from point x to point x' at time t. Thus, the intrinsic convolution is defined as:

$$\left(f*g\right)(x)=\int g\left(\theta,t\right)\left(D_\alpha\left(x\right)f\right)\left(\theta,t\right)dtd\theta \tag{37}$$

Unlike the arbitrarily oriented geodesic patches in GCNN, ACNN's construction mainly uses the principal curvature direction as the reference $\theta=0$. The most promising future work direction is applying ACNN to learning on graphs. Both the GCNN and ACNN methods operate in spatial domains, avoid the inherent drawbacks of standardizing the spectral methods with different domains, and have proved to be more adept at looking for deformable shapes than traditional hand-crafted methods.

Mixture model network (MoNet) [23]: In [23], the author proposed an unified framework that can extend CNN structure to non-Euclidean domains (graphs and manifolds). Because some spatial-based methods ignore the relative positions between nodes and their neighbors when integrating information of neighbor nodes. Therefore, MoNet introduces pseudo-coordinate system and weight function, so that the weight of node neighbors depends on the relative position of node and its neighbor nodes. MoNet can learn local, stationary and combinatorial specific task characteristics, which also indicates that various non-Euclidean CNN methods proposed in previous literatures can be regarded as a special case of MoNet.

MoNet framework uses x to denote a point on a manifold or a vertex of a graph, and consider points $x'\in N(x)$ in the neighborhood of x. For each x, correlate a d-dimensional vector of pseudo-coordinates $\mathbf{u}\left(x,x'\right)$. MoNet constructs the patch operator as the following general form:

$$D_j(x)f=\sum_{x\in N(x)} w_j\left(\mathbf{u}\left(x,x'\right)\right)f\left(x'\right),\quad j=1,\cdots,J \tag{38}$$

where $\mathbf{W}_\Theta(\mathbf{u})=(w_1(\mathbf{u}),\cdots,w_J(\mathbf{u}))$ is a weighting function (kernel) parametrized by learnable parameters $\mathbf{\Theta}$, and each Gaussian kernel is defined as $w_j(\mathbf{u})=\exp\left(-\frac{1}{2}\left(\mathbf{u}-\boldsymbol{\mu}_j\right)^{\mathrm{T}}\sum_j^{-1}\left(\mathbf{u}-\boldsymbol{\mu}_j\right)\right)$ in which parameters $d\times d$ covariance matrix $\sum_1,\cdots,\sum_J$ and $d\times 1$ mean vectors $\boldsymbol{\mu}_1,\cdots,\boldsymbol{\mu}_J$ are learnable and J represents the dimensionality of the extracted patch. Thus, the generalization form of the spatial convolution on non-Euclidean domains is given by using a template-matching process:

$$(f*g)(x)=\sum_{j=1}^{J} g_j D_j(x)f \tag{39}$$

The two key choices in this construction are the pseudo-coordinates $\mathbf{u}$ and the weight functions $\mathbf{W}_\Theta(\mathbf{u})$. Different definition of $\mathbf{u}$ and $\mathbf{W}_\Theta(\mathbf{u})$ produces different deep learning methods, including the classical CNN on Euclidean domains, GCN on graphs, and GCNN and ACNN on manifolds. For example, GCNN and ACNN use Gaussian kernels for local polar geodesic coordinates on maniflod, and GCN applies triangular kernel on the degree of the graph vertices' pseudo-coordinates. This shows that all the non-Euclidean CNN methods proposed in previous literatures can be regarded as a special case of MoNet.

Structured Prediction Model (FMNet) [22]: Litany *et al.* proposed dense shape correspondence based on neural

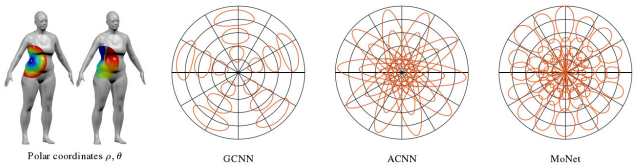

FIGURE 7. Left: intrinsic local polar coordinates ρ, θ on manifold around a point marked in white. Right: patch operator weighting functions $w_i(\rho, \theta)$ used in different generalizations of convolution on the manifold. Cited from [23].

network, which is a task-driven approach for descriptor learning, by including the computation of the correspondence directly as part of the learning procedure. This is a sharp contrast to the former descriptor learning technique, which does not consider post-processing during training.

Existing learning-based methods take 3D shape correspondence as a mapping problem, in which each point of the query shape receives a label that identifies a point in a reference shape. Then, the correspondence is the a posteriori by predicting the labels of the two input shapes. The authors used deep residual networks to model the learning process to process dense descriptors fields defined on the two shapes to provide compact representation of the correspondence.

5) COMBINED SPECTRAL / SPATIAL METHOD

The fourth construction of convolution-like operation on non-Euclidean domains is jointly in spatial-frequency domain.

Localized Spectral CNN (LSCNN) [21]: In the field of classical signal processing, the exploration of space localization has been a major defect of classical Fourier analysis. However, this problem can be well solved by frequency analysis on Windows, that is, Fourier transform of Windows [70] (WFT, also known as short-time Fourier transform or spectrogram in signal processing). In [21], Boscaini *et al.* proposed an intrinsic CNN construction (local spectral CNN) based on the Fourier transform of Windows, extended WFT to manifold and point cloud data, and extracted the local behavior of some dense intrinsic descriptors, roughly acting as an analogy to patches in images. The local frequencies can be obtained by constructing patch operator in geometric data and applying learnable filters to such patches, and the subsequent work is similar to that in CNN, that is, the coefficients of the filter are constantly updated by minimizing the cost function. It should be noted that the weights, windows coefficients and filters of all layers are all variables discovered through supervised learning. This method is an extension of the existing graph-based spectral CNN [12], and allows learning class-specific shape descriptors.

LSCNN's fully connected layer produces Q-dimensional outputs as weighted (w_{qp} as coefficients) sums of the P-dimensional inputs, followed by a non-linear ReLU activation function. Theoretically, we can use any intrinsics spectral descriptor as the input to this layer.

LSCNN's convolutional layer applies the WFT to extract the local structure of the Q-dimensional input around each point. Since each input dimension might contain features of different scale, we employ different window for each input dimension. The family of Q windows is parametrized in some fixed interpolation basis in the frequency domain where the $Q \times M$ matrix b_{pm} of weights defines the windows. The WFTs are then passed through a bank of filters applied in frequency domain, producing the outputs used as the R-dimensional final LSCNN descriptor.

The LSCNN is a parametric hierarchical system producing a R-dimensional descriptor at each point x. Its loss function aims to estimate the optimal task-specific parameters of the descriptor minimizing the aggregate loss. The parameters to learn include w_{qp}, b_{pm} and a_{qr}.

One of the main advantages of LSCNN is that the same framework can be applied to different shape representations, especially grids and point clouds. However, one disadvantage of this approach is the large memory usage and intensive computation, since each window needs to be explicitly generated.

6) POINT-BASED METHOD

Point based method can directly use point cloud data as input without converting to voxel, mesh and other 3D representations, where a point cloud is a set of unordered vectors. Unlike voxel-based and multi-view-based methods, methods using point cloud directly process the spatial coordinates of the original 3D graphics surface and must be invariant to the number of data points and the data order or permutation. The first work is the PointNet [64] framework released by Charles R. Qi of Stanford university in 2017. Then at the end of 2017, the author proposed the PointNet++ [71] framework by imitating the hierarchical feature extraction idea of CNN in the original framework, which further improved the accuracy of classification and segmentation tasks.

PointNet [64]: PointNet is the first deep neural network that can directly process the 3D point cloud. The basic idea of PointNet is to learn the corresponding spatial coding of each point in the input point cloud, and then use the features of all points to get a point cloud's global feature. In this work, the author takes the max-pooling layer as a symmetric function to aggregate information from all the points to deal with the disordered nature of the point cloud model, that is to say, no matter how the input order is, the output will be the same by max-pooling. It is easy to find that sum-pooling, average-pooling operation can achieve the same results.

The input is n sample points, each with x, y, z coordinates. After several steps of MLP (multi-layer perceptron), at each point, features of 1024 dimensions are obtained, and then a symmetric operation which is max pooling is used to produce the global feature of 1024 dimensions. The author designed two "T-net" networks and used a rotation matrix to deal with rotation invariance of the model. The input transform in first T-net is to adjust the point cloud in the space. The feature transform in second T-net is to align the extracted 64 dimensional features, that is, to transform the point cloud at the feature level. For classification task, the global features is passed through MLP to generate the softmax classification

probability, and for segmentation task, the global features are concatenated with the features from individual points before input into the MLP for the final classification of each data point.

PointNet++ [71]: The author proposed an improved version of PointNet++ to solve two problems faced by PointNet, namely, how to carry out local division of point cloud and how to carry out local feature extraction of point cloud. Its core idea is similar to the convolutional neural network. PointNet++ is essentially a hierarchical version of PointNet. Feature extraction of each subpoint set in the hierarchical point cloud consists of three parts:

1) Sampling layer: the sampling layer selects a series of points from the input point cloud, from which centroid points for the local region are defined, and the sampling algorithm uses iterative farthest point sampling (FPS) method;
2) Grouping layer: the grouping layer aims to construct local regions and then extract features, i.e., multiple subpoint clouds are created by using neighboring points around centroid points (within a given radius), and neighborhood ball rather than KNN is used in this paper because a fixed region scale can be guaranteed;
3) PointNet layer: to extract local features of point cloud, the original PointNet can be used. So in PointNet++, the original PointNet network becomes a sub-network in PointNet++ network, hierarchical iteration extraction features.

In 3D scanning, it is very likely that the sampling density of points are not uniformly distributed due to factors like perspective effects, radial density variations, motions etc, which poses a significant challenge for feature learning based on point cloud. To solve this problem, the authors proposed abstraction layers in the model to aggregate multi-scale information according to local point densities. PointNet++ achieved 90.7% accuracy on ModelNet40 [66], which was the state-of-the-art.

7) GEOMETRIC ALGEBRA-BASED METHOD

William K. Clifford introduced Geometric Algebra abbreviated as GA, also called Clifford Algebra, which provides such a coordinate-free framework to make the computation efficiently. It completes the constructions and modelings in a coordinate-free way and also has revealed wide applications especially when applying to computer vision tasks, such as multispectral images reconstruction or multispectral image denoising [72] and 3D geometrical shapes classification [73].

RGA-MLP [73]: Multilayer Perceptron (MLP) is an efficient feed-forward neural network, constituting one of the most common and popular classes of neural networks for image processing and pattern recognition. It consists of several subsequent layers which is of perceptron-type, including an input layer that simply obtains the external inputs, a set of hidden layers and one output layer. And Geometric Algebra (GA) is introduced by William K. Clifford, also known as Clifford Algebra, which gives geometric insight and effective representation for multi-dimensional signals.Technically speaking, GA subsume, for example, the real numbers, the complex numbers and the quaternions.

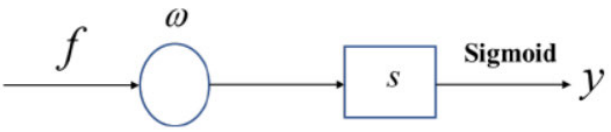

FIGURE 8. The structure of RGA neuron model.Cited from [73].

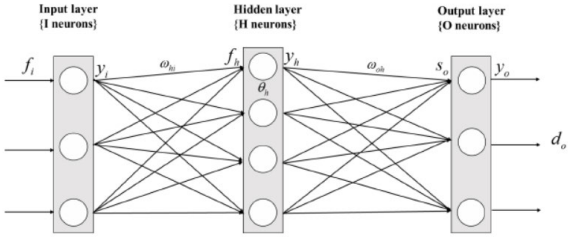

FIGURE 9. The structure of RGA-MLP model with one hidden layer. Cited from [73].

The authors Y. P. Li and W. M. Cao proposed a reduced version of GA called RGA and utilize RGA to obtain an extension neuron model for multi-dimensional signal processing, in which all the operators can be extended to RGA domain. The traditional multi-dimensional signal MLP model treats the multi-dimensional signal as real number and processes it independently. By contrast, they presented a traditional multi-layer perceptron (MLP) extension model for multi-dimensional signal processing from real numbers to RGA domain by using proposed RGA neurons. The structures of RGA neuron and RGA-MLP model are showed below.

The network for 3D geometrical shapes task is a three layer network, which has one hidden layer. Models are optimized with learning rate set at 0.04. The training ends at iteration 7200. The results illustrate that the RGA-MLP model outperforms the traditional real-valued MLP. It can be seen that multi-dimensional signals processed by real-valued MLP tends to be single channel independently, which are not rich enough to preserves more discriminative information among multiple channels. In contrast, the proposed RGA-MLP model is capable of capturing the inter-relationship information between different channels, such as the scaling and the rotation of inputs in multi-dimensional space, this information is more important for classification. For more details in geometric algebra-based methods, please read the revelant technical paper [74]–[76], especially a comprehensive review [77] in this area.

IV. APPLICATIONS

In this section, we will first introduce some basic datasets which are commonly used in the geometric deep learning field, including graph network datasets and some 3D model

datasets. Then list several practical applications of geometric deep learning.

A. DATASETS

1) NON-STRUCTURAL DATASET

Images,text and videos are most common unstructured data. There are some common datasets such as **MNIST** [78], **ImageNet** [79] and **20NewsGroup** [80].

The **MNIST** dataset contains 70,000 images of size 28×28 labeled with ten digits. A typical way to convert an MNIST image to a graph is to construct an 8-NN graph based on its pixel locations.

The **ImageNet** dataset is a large image database used for visual object recognition, in which over 14 million images are manually annotated and at least a million images, bounding boxes for objects are also provided. ImageNet contains more than 20,000 categories and acts like a network with multiple nodes, each of which is equivalent to an item or subcategory.

The **20NewsGroup** dataset consists of around 20,000 News Group (NG) text documents categorized by 20 news types. The graph of the 20NewsGroup is constructed by representing each document as a node and using the similarities between nodes as edge weight.

2) CITATION NETWORK DATASET

The citation network is a collection composed of inter-literature references and quotation relationships in which nodes represent papers and edges represent citation relations. There are four popular datasets for paper-citation networks, such as **Cora** [81], **Citeseer** [82], **WebKB** from https://starling.utdallas.edu/datasets/webkb/ and **Aminer Author-Paper-Citation (Aminer APC) Network** from https://aminer.org/citation.

The **Cora** dataset consists of 2,708 scientific publications classified into seven classes and 5,429 links.

The **CiteSeer** dataset consists of 3,312 scientific publications classified into six classes and 4,732 links.

The **WebKB** dataset consists of 877 scientific publications classified into five classes and 1,608 links. Each publication in Cora, Citeseer and WebKB is described by a 0/1-valued word vector indicating the absence or presence of the corresponding word from the dictionary.

The **APC Network** is a large-scale and complete academic dataset released by Arnetminer.org which includes 2,092,356 papers 8,024,869 citations between papers 1,712,433 authors and 4,258,615 collaboration relationships.

3) SOCIAL NETWORK DATASET

A social network is a network describing the interactions between people. It is an online social net, each node represents a user and edges representing interactions between people. There are five popular datasets such as **Facebook**, **Gplus**, **Twitter**, **Reddit** and **Epinions1**. You can download these data from http://snap.stanford.edu/data.

The **Facebook** dataset is an undirected graph which consists of 'circles' (or 'friends lists') from Facebook. Facebook data was collected from survey participants using Facebook app. The dataset includes node features (profiles), circles, and ego networks.

The **Gplus** dataset is a directed graph which consists of 'circles' from Google+. Google+ data was collected from users who had manually shared their circles using the 'share circle' feature. The dataset also includes node features (profiles), circles, and ego networks.

The **Twitter** dataset is a directed graph consisted of 'circles' (or 'lists') from Twitter. Twitter data was crawled from public sources.

The **Reddit** dataset is a collection of 132,308 reddit.com submissions. For each submission, we collect features such as the number of ratings (positive/negative), the submission title, and the number of comments it received.

The **Epinions1** dataset is a who-trust-whom online social network of Epinions.com which is a general consumer review site. Members of the site can decide whether to ''trust'' each other. All the trust relationships interact and form the trust network, which is combined with review ratings to determine which review should be presented to users.

4) CHEMISTRY / BIOLOGY DATASET

Chemical molecules and compounds can be represented by chemical graphs with atoms as nodes and chemical bonds as edges. This category of graphs is often used to evaluate graph classification performance.

The **NCI-1** and **NCI-9** dataset contain 4,110 and 4,127 chemical compounds respectively, labeled as whether they are active to hinder the growth of human cancer cell lines. The datasets can be found on http://networkrepository.com/NCI1.php and http://featureselection.asu.edu/datasets.php.

The **MUTAG** [83] dataset contains 188 nitro compounds, labeled as whether they are aromatic or heteroaromatic.

The **D&D** dataset contains 1,178 protein structures, labeled as whether they are enzymes or non-enzymes. The dataset can be found on https://github.com/snap-stanford/GraphRNN/tree/master/dataset/DD.

The **QM9** [84] dataset contains 133,885 molecules labeled with 13 chemical properties.

The **Tox21** [85] dataset contains 12,707 chemical compounds labeled with 12 types of toxicity.

Another important dataset is the **Protein-Protein Interaction network (PPI)**, containing 24 biological graphs with nodes represented by proteins and edges represented by the interactions between proteins. The datasets can be found on https://genemania.org/. In PPI, each graph is associated with one human tissue. Each node is labeled with its biological states.

5) TRAFFIC NETWORK DATASET

There are several datasets of the traffic conditions. The **METR-LA** [86] is a traffic dataset collected from the highways of Los Angeles County.

The **Roadnet-CA** dataset is a road network of California. Intersections and endpoints are represented by nodes and the roads connecting these intersections or road endpoints are represented by undirected edges. The datasets can be found on http://snap.stanford.edu/data/roadNet-CA.html.

6) 3D SHAPE DATASET

There is an increasing interest in working with 3D geometric data, due to the emergence of 3D data collected by VR (visual reality) sensors or Microsoft Kinect. Most 3D shapes are modeled as Riemannian manifolds and discretized as meshes.

The **FAUST** [87] dataset (Fine Alignment Using Scan Texture) contains 300 high-resolution human body scans of 10 different subjects in 30 different poses. Scans are acquired through a 3D multi-stereo system, each of which is represented as a high-resolution, triangulated, non-waterlight mesh.

The **TOSCA** dataset is used for high-resolution 3D non-rigid shapes in a variety of poses for non-rigid shape similarity and correspondence experiments. It contains a total of 80 objects, including 11 cats, 9 dogs, 3 wolves, 8 horses, 6 centaurs, 4 gorillas, 12 female figures, and 2 different male figures, containing 7 and 20 poses. Typical vertex count is about 50,000. Turnable is 3D object dataset in multi-viewed form from 144 calibrated viewpoints under 3 different lighting conditions. The objects are placed on an automated turntable and photographed every 5 degrees. And each object is photographed by 2 cameras arranged in a stereo setup to obtain images with a resolution of 3 million pixels. This dataset can be found on http://tosca.cs.technion.ac.il/book/resources_data.html.

B. PRACTICAL APPLICATIONS

So far, we have introduced the knowledge of geometric deep learning, reviewed the deep learning methods based on graphs and manifold data, and some datasets that are often used in this field. Next, we will introduce some practical applications in this field.

1) COMPUTER VISION

One of the biggest applications of deep geometric learning is computer vision. The most common application examples in geometric deep learning are network analysis. In general, the data is embodied by a graph where the vertices denote members such as paper or users, and the relationship between members are represented by the directed or undirected edges. Classification task is to assign each data member to a class. For example, work [16], [23] applied the spectral-based CNN with two spectral convolutional layers on Cora dataset. The work [88] worked on a social influence network.

3D data analysis has received more and more interest from research scholars. In this domain, 3D shape data is modeled as Riemannian manifolds and discretized as meshes. There are two main application directions. One is to learn local descriptors [18], [60] and shape correspondence [19]. Another direction is to learn global descriptors and use them in shape recognition tasks [18]. Recognition of human motion in videos can help better understand video content from the machine perspective. In the work of [89], [90], the authors used spatial-temporal neural networks to learn human motion pattern by examining the time series of human skeleton in the video clip.

In addition, possible directions for applying graph neural network in computer vision continues to grow, including small-sample image classification [91], semantic segmentation [92], visual reasoning [93], and QA systems [94].

2) RECOMMENDER SYSTEMS

The recommender systems are widely used in movie website (Netflix), social software (WeChat) and online shopping platform. Online shopping sites usually build network or graph using goods or customers as the nodes. This data processing mode has become ubiquitous in transaction management by utilizing the relationship between goods and goods, customers and customers, customers and goods. The key to a recommender system is to rate the importance of an item to the user, which can be converted into a link prediction problem. The goal is to predict the missing link between the users and the items. To address this problem, the work in [95] proposed an auto-encoder model based on graphs. Monti *et al.* [26] combined GCN with RNN framework to learn the hidden procedure of generating the known ratings.

3) BIOLOGY/CHEMISTRY

In chemistry, researchers have mimicked GNN's core ideas to study the structure of molecular graphs. In molecular graphs, nodes represent atoms and edges represent bonds. Wilensky, Uri, and Kenneth Reisman describe a computation-based approach that enables students to investigate the connections between different biological levels [96]. The node classification, graph classification, and graph generation are the three main tasks, which can learn molecular fingerprints [24], predict molecular properties [97], infer protein interfaces [98], and synthesize chemical compounds [99].

4) OTHERS

At present, some preliminary exploration have been made to apply geometric deep learning to other problems such as traffic flow prediction [100], brain functional networks detection [101], physics [102], disease or drug prediction [103], [104], and natural language processing [105].

If readers are engaged in relevant research with spectral methods for fixed graphs, practical applications in this direction include citation/social networks, recommender systems, medical imaging, etc. If you study spatial methods for arbitrary graphs, related applications include particle physics and chemistry, molecule design, quadratic the assignment problem, and so on. If readers do the research of spatial methods for manifolds, related applications include 3D shape correspondence / classification and so on. In addition, if you are interested, you can follow the work of Charles R. Qi to further study the related technologies of point cloud.

V. FUTURE WORK AND OPEN PROBELMS

Although geometric deep learning technique has proved itself in learning both Euclidean and non-Euclidean domains data, especially in non-Euclidean domain, there still exist some challenges and problems to solve in this field.

A. COMPUTATION COST

Though the graph neural network has achieved great success in various fields, the computation cost still a problem for researchers and applications. Deep learning neural network has a large number of parameters, which makes the computation intensive in training and test stage. The same is true for graph neural networks. Moreover, due to the complex relationship between graph nodes and the non-grid nature, the computation is even more heavy. On the other hand, most existing deep learning frameworks deal with regularized structured data in Euclidean domain, for example, a one-dimensional or two-dimensional grid which are allowed to take the advantages of powerful processing ability of modern GPUs. But the geometric data does not lie in a grid-like structure in most cases, so they need different methods to realize efficient and complex computation. So, how to make graph neural network run faster is an urgent need.

B. DEEPER ARCHITECTURE

In the deep learning field, the CNN architecture has made impressive achievements in the problem of Euclidean data. As the number of network layers increases, the model becomes more complex. Empirically, neural networks that use more parameters tend to have better performance.

But stacked multi-layered GNNS will raise the over-smoothing problem. Unlike the independent pixels in the images, the nodes in the graph interact with neighbors and the graph process in the network is a flow of information diffusion and aggregation. The more layers to stack, information from nodes will be combined together which will lead the same representation of all nodes in the end. Therefore, all the vertices will converge to the same value. For example, most of the advanced graph convolutional network frameworks are no more than 3 or 4 layers. The work of [106] tried to use deeper network structure, but the performance is not satisfactory. However, in 2019, Deepgcns [107] successfully built a network of 56 layers by borrowing the ideas of residual / dense connections and dilated convolutions. Designing deep graph neural network is promising, although still a challenging problem.

C. DYNAMIC GRAPHS

Most current methods are used to work with static graphs, and many datasets are also static. In real life, however, many graphs changes over time. For example, the exit of existing users and the addition of new users in social networks may occur at any time, and the relationship between users may change with time. How to effectively model the evolution of dynamic graph is still an unsolved problem, which affects the practicability of graph neural network to some extent. There are some attempts to solve this problem, such as [108]–[110].

D. SCALABILITY / GENERALIZATION

It is a difficult problem to apply the graph neural network on large graph. On the one hand, each node has its own neighborhood structure, which involves the hidden state of neighboring nodes, so it is difficult to use batch method in training [29]. On the other hand, when dealing with millions of nodes and edges, the researchers found it difficult to calculate the Laplacian matrix of the graph. There are also some methods to improve the model's performance through rapid sampling [111], [112] and subgraph training [46], [47] methods, but the results are not very good.

E. CAUSAL REASONING

It is exciting that the graph neural network has the potential to solve the problem pointed out by Turing award winner Judea Pearl that deep learning cannot do causal reasoning [113]. The graph neural network is the generalization and expansion of various neural network methods of graph before which has powerful relational inductive bias that provides a direct interface for manipulating structured knowledge and generating structured behaviors, and combines end-to-end learning with inductive reasoning.

It is expected to solve the problem that deep learning cannot carry out relational reasoning [114]. Relevant work includes [115], which takes the first step and focuses on using a general framework of GNNs and GCNs for relational solving problems.

VI. CONCLUSION

Thanks to the easy availability of computing resources and dataset, as well as the rapid development in deep learning technologies from Euclidean domain such as texts, images and videos, geometric deep learning (GDL) methods has achieved great success in theoretical research and applications. In this review, we have comprehensively reviewed deep learning methods in the graph and manifold domain, from the history, background to the methods, including deep networks on graphs and manifold. Following that, typical applications, benchmark datasets, and existing problems and challenges are discussed. We believe this review will provide a good reference for the readers and researchers in this domain.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258.
[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.
[4] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.
[7] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.
[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[9] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2014–2023.
[10] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2, Aug. 2005, pp. 729–734.
[11] F. Scarselli, M. Gori, A. Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.
[12] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*. [Online]. Available: http://arxiv.org/abs/1312.6203
[13] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, *arXiv:1506.05163*. [Online]. Available: http://arxiv.org/abs/1506.05163
[14] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1993–2001.
[15] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
[16] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*. [Online]. Available: http://arxiv.org/abs/1609.02907
[17] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Trans. Signal Process.*, vol. 67, no. 1, pp. 97–109, Jan. 2019.
[18] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on riemannian manifolds," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop (ICCVW)*, Dec. 2015, pp. 37–45.
[19] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning shape correspondence with anisotropic convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3189–3197.
[20] L. Yi, H. Su, X. Guo, and L. Guibas, "SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2282–2290.
[21] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst, "Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks," *Comput. Graph. Forum*, vol. 34, no. 5, pp. 13–23, Aug. 2015.
[22] O. Litany, T. Remez, E. Rodola, A. Bronstein, and M. Bronstein, "Deep functional maps: Structured prediction for dense shape correspondence," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5659–5667.
[23] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5115–5124.
[24] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2224–2232.
[25] Z. Huang, C. Wan, T. Probst, and L. V. Gool, "Deep learning on lie groups for skeleton-based action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6099–6108.
[26] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3697–3707.
[27] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.
[28] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," 2018, *arXiv:1812.04202*. [Online]. Available: http://arxiv.org/abs/1812.04202
[29] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," 2018, *arXiv:1812.08434*. [Online]. Available: http://arxiv.org/abs/1812.08434
[30] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," 2019, *arXiv:1901.00596*. [Online]. Available: http://arxiv.org/abs/1901.00596
[31] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, May 2013.
[32] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.
[33] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, Apr. 1980.
[34] W. Cao, Q. Lin, Z. He, and Z. He, "Hybrid representation learning for cross-modal retrieval," *Neurocomputing*, vol. 345, pp. 45–57, Jun. 2019.
[35] D. Meng, L. Zhang, G. Cao, W. Cao, G. Zhang, and B. Hu, "Liver fibrosis classification based on transfer learning and FCNet for ultrasound images," *IEEE Access*, vol. 30, pp. 5804–5810, 2017.
[36] W. Cao, W. Feng, Q. Lin, G. Cao, and Z. He, "A review of hashing methods for multimodal retrieval," *IEEE Access*, vol. 8, pp. 15377–15391, 2020.
[37] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 2843–2851.
[38] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," 2013, *arXiv:1312.6229*. [Online]. Available: http://arxiv.org/abs/1312.6229
[39] W. Cao, J. Yuan, Z. He, Z. Zhang, and Z. He, "Fast deep neural networks with knowledge guided training and predicted regions of interests for real-time video object detection," *IEEE Access*, vol. 6, pp. 8990–8999, 2018.
[40] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 664–676, Apr. 2017.
[41] Z. Zhang, M. Li, X. Lin, Y. Wang, and F. He, "Multistep speed prediction on traffic networks: A deep learning approach considering spatio-temporal dependencies," *Transp. Res. C, Emerg. Technol.*, vol. 105, pp. 297–322, Aug. 2019.
[42] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *Proc. 32nd AAAI Conf. Artif. Intell.*, Apr. 2018, pp. 1–8.
[43] J. Yan, C. Li, Y. Li, and G. Cao, "Adaptive discrete hypergraph matching," *IEEE Trans. Cybern.*, vol. 48, no. 2, pp. 765–779, Feb. 2018.
[44] L. B. Almeida, "A learning rule for asynchronous perceptrons with feedback in a combinatorial environment," in *Proc. IEEE 1st Int. Conf. Neural Netw.*, vol. 2, Jan. 1987, pp. 609–618.
[45] F. J. Pineda, "Generalization of back-propagation to recurrent neural networks," *Phys. Rev. Lett.*, vol. 59, no. 19, pp. 2229–2232, Nov. 1987.
[46] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.
[47] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2018, pp. 1416–1424.
[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[49] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*. [Online]. Available: http://arxiv.org/abs/1710.10903
[50] J. B. Lee, R. Rossi, and X. Kong, "Graph classification using structural attention," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2018, pp. 1666–1674.
[51] B. Chen, L. Sun, and X. Han, "Sequence-to-action: End-to-End semantic graph generation for semantic parsing," 2018, *arXiv:1809.00773*. [Online]. Available: http://arxiv.org/abs/1809.00773
[52] D. D. Johnson, "Learning graphical state transitions," in *Proc. ICLR*, 2016, pp. 1–19.
[53] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating graphs via random walks," 2018, *arXiv:1803.00816*. [Online]. Available: http://arxiv.org/abs/1803.00816

[54] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," 2018, *arXiv:1802.08773*. [Online]. Available: http://arxiv.org/abs/1802.08773
[55] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*. [Online]. Available: http://arxiv.org/abs/1611.07308
[56] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," 2018, *arXiv:1802.04407*. [Online]. Available: http://arxiv.org/abs/1802.04407
[57] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2016, pp. 1225–1234.
[58] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network embedding with regular equivalence," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2018, pp. 2357–2366.
[59] R. Litman and A. M. Bronstein, "Learning spectral descriptors for deformable shape correspondence," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 171–180, Jan. 2014.
[60] D. Boscaini, J. Masci, E. Rodolà, M. M. Bronstein, and D. Cremers, "Anisotropic diffusion descriptors," *Comput. Graph. Forum*, vol. 35, no. 2, pp. 431–441, May 2016.
[61] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov, "Shape Google: Geometric words and expressions for invariant shape retrieval," *ACM Trans. Graph.*, vol. 30, no. 1, pp. 1–20, Jan. 2011.
[62] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.
[63] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 945–953.
[64] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.
[65] C. R. Qi, H. Su, M. NieBner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view CNNs for object classification on 3D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5648–5656.
[66] *ModelNet*. Accessed: Oct. 2019. [Online]. Available: https://modelnet.cs.princeton.edu/
[67] J. Sun, M. Ovsjanikov, and L. Guibas, "A concise and provably informative multi-scale signature based on heat diffusion," *Comput. Graph. Forum*, vol. 28, no. 5, pp. 1383–1392, Jul. 2009.
[68] M. Aubry, U. Schlickewei, and D. Cremers, "The wave kernel signature: A quantum mechanical approach to shape analysis," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCV Workshops)*, Nov. 2011, pp. 1626–1633.
[69] I. Kokkinos, M. M. Bronstein, R. Litman, and A. M. Bronstein, "Intrinsic shape context descriptors for deformable shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 159–166.
[70] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *Appl. Comput. Harmon. Anal.*, vol. 40, no. 2, pp. 260–291, Mar. 2016.
[71] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.
[72] R. Wang, M. Shen, and W. Cao, "Multivector sparse representation for multispectral images using geometric algebra," *IEEE Access*, vol. 7, pp. 12755–12767, 2019.
[73] Y. Li and W. Cao, "An extended multilayer perceptron model using reduced geometric algebra," *IEEE Access*, vol. 7, pp. 129815–129823, 2019.
[74] E. Hitzer, "Geometric operations implemented by conformal geometric algebra neural nodes," 2013, *arXiv:1306.1358*. [Online]. Available: http://arxiv.org/abs/1306.1358
[75] W. Cao, F. Lyu, Z. He, G. Cao, and Z. He, "Multimodal medical image registration based on feature spheres in geometric algebra," *IEEE Access*, vol. 6, pp. 21164–21172, 2018.
[76] R. Wang, Z. Cao, X. Wang, W. Xue, and W. Cao, "GA-STIP: Action recognition in multi-channel videos with geometric algebra based spatio-temporal interest points," *IEEE Access*, vol. 6, pp. 56575–56586, 2018.
[77] E. Hitzer, T. Nitta, and Y. Kuroe, "Applications of clifford's geometric algebra," *Adv. Appl. Clifford Algebras*, vol. 23, no. 2, pp. 377–404, 2013.
[78] Y. LeCun, C. Cortes, and C. J. Burges. (1998). *The MNIST Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist
[79] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[80] K. Albishre, M. Albathan, and Y. Li, "Effective 20 newsgroups dataset cleaning," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol. (WI-IAT)*, Dec. 2015, pp. 98–101.
[81] A. McCallum, "Cora dataset," LINQS, Inter-Univ. Consortium Political Social Res., Santa Cruz, CA, USA, Tech. Rep., 2017.
[82] C. Caragea, J. Wu, A. Ciobanu, K. Williams, J. Fernández-Ramírez, H.-H. Chen, Z. Wu, and L. Giles, "Citeseer x: A scholarly big dataset," in *Proc. Eur. Conf. Inf. Retr.* New York, NY, USA: Springer, 2014, pp. 311–322.
[83] A. K. Debnath, R. L. Lopez de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch, "Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity," *J. Medicinal Chem.*, vol. 34, no. 2, pp. 786–797, Feb. 1991.
[84] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," *Sci. Data*, vol. 1, no. 1, Aug. 2014, Art. no. 140022.
[85] R. R. Tice, C. P. Austin, R. J. Kavlock, and J. R. Bucher, "Improving the human hazard characterization of chemicals: A Tox21 update," *Environ. Health Perspect.*, vol. 121, no. 7, pp. 756–765, Jul. 2013.
[86] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," 2017, *arXiv:1707.01926*. [Online]. Available: http://arxiv.org/abs/1707.01926
[87] F. Bogo, J. Romero, M. Loper, and M. J. Black, "FAUST: Dataset and evaluation for 3D mesh registration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* Piscataway, NJ, USA: IEEE, Jun. 2014, pp. 3794–3801.
[88] J. Qiu, J. Tang, H. Ma, Y. Dong, K. Wang, and J. Tang, "Deepinf: Modeling influence locality in large social networks," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2018, pp. 1–9.
[89] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–9.
[90] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep learning on spatio-temporal graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 5308–5317.
[91] M. Guo, E. Chou, D.-A. Huang, S. Song, S. Yeung, and L. Fei-Fei, "Neural graph matching networks for fewshot 3D action recognition," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 653–669.
[92] X. Qi, R. Liao, J. Jia, S. Fidler, and R. Urtasun, "3D graph neural networks for RGBD semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5199–5208.
[93] X. Chen, L.-J. Li, L. Fei-Fei, and A. Gupta, "Iterative visual reasoning beyond convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7239–7248.
[94] M. Narasimhan, S. Lazebnik, and A. Schwing, "Out of the box: Reasoning with graph convolution nets for factual visual question answering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 2654–2665.
[95] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for Web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2018, pp. 974–983.
[96] U. Wilensky and K. Reisman, "Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational Theories—An embodied modeling approach," *Cognition Instruct.*, vol. 24, no. 2, pp. 171–209, Jun. 2006.
[97] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn. (JMLR)*, vol. 70, 2017, pp. 1263–1272.
[98] A. Fout, J. Byrd, B. Shariat, and A. Ben-Hur, "Protein interface prediction using graph convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6530–6539.
[99] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," 2018, *arXiv:1805.11973*. [Online]. Available: http://arxiv.org/abs/1805.11973

[100] Z. Cui, K. Henrickson, R. Ke, Z. Pu, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," 2018, *arXiv:1802.07007*. [Online]. Available: http://arxiv.org/abs/1802.07007
[101] S. I. Ktena, S. Parisot, E. Ferrante, M. Rajchl, M. Lee, B. Glocker, and D. Rueckert, "Distance metric learning using graph convolutional networks: Application to functional brain networks," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* New York, NY, USA: Springer, 2017, pp. 469–477.
[102] C. W. Coley, R. Barzilay, W. H. Green, T. S. Jaakkola, and K. F. Jensen, "Convolutional embedding of attributed molecular graphs for physical property prediction," *J. Chem. Inf. Model.*, vol. 57, no. 8, pp. 1757–1772, Jul. 2017.
[103] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, Jun. 2018.
[104] S. Parisot, S. I. Ktena, E. Ferrante, M. Lee, R. G. Moreno, B. Glocker, and D. Rueckert, "Spectral graph convolutions for population-based disease prediction," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.* New York, NY, USA: Springer, 2017, pp. 177–185.
[105] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," 2017, *arXiv:1704.04675*. [Online]. Available: http://arxiv.org/abs/1704.04675
[106] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
[107] G. Li, M. Muller, A. Thabet, and B. Ghanem, "DeepGCNs: Can gcns go as deep as CNNs?" in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9267–9276.
[108] M. Looks, M. Herreshoff, D. Hutchins, and P. Norvig, "Deep learning with dynamic computation graphs," 2017, *arXiv:1702.02181*. [Online]. Available: http://arxiv.org/abs/1702.02181
[109] Y. Ma, Z. Guo, Z. Ren, E. Zhao, J. Tang, and D. Yin, "Streaming graph neural networks," 2018, *arXiv:1810.10627*. [Online]. Available: http://arxiv.org/abs/1810.10627
[110] F. Manessi, A. Rozza, and M. Manzo, "Dynamic graph convolutional networks," *Pattern Recognit.*, vol. 97, Jan. 2020, Art. no. 107000.
[111] J. Chen, J. Zhu, and L. Song, "Stochastic training of graph convolutional networks with variance reduction," 2017, *arXiv:1710.10568*. [Online]. Available: http://arxiv.org/abs/1710.10568
[112] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," 2018, *arXiv:1801.10247*. [Online]. Available: http://arxiv.org/abs/1801.10247
[113] J. Pearl, "Theoretical impediments to machine learning with seven sparks from the causal revolution," 2018, *arXiv:1801.04016*. [Online]. Available: http://arxiv.org/abs/1801.04016
[114] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4967–4976.
[115] P. W. Battaglia *et al.*, "Relational inductive biases, deep learning, and graph networks," 2018, *arXiv:1806.01261*. [Online]. Available: http://arxiv.org/abs/1806.01261

WENMING CAO (Member, IEEE) received the M.S. degree from the System Science Institute, China Science Academy, Beijing, China, in 1991, and the Ph.D. degree from the School of Automation, Southeast University, Nanjing, China, in 2003. From 2005 to 2007, he was a Postdoctoral Researcher with the Institute of Semiconductors, Chinese Academy of Sciences. He is currently a Professor with Shenzhen University, Shenzhen, China. He is also a Project Manager of The Guangdong Key Laboratory of Intelligent Information Processing, Guangdong Multimedia Information Service Engineering Technology Research Center and Video Processing and Communication Lab, Department of Electrical and Computer Engineering. He has authored or coauthored more than 80 publications in top-tier conferences and journals. His research interests include pattern recognition, image processing, and visual tracking.

ZHIYUE YAN was born in Taiyuan, Shanxi, China, in 1995. He received the B.S. degree in information engineering from Shenzhen University, Shenzhen, Guangdong, China, in 2018, where he is currently pursuing the M.S. degree in information engineering with the College of Electronic and Information Engineering. His research interests include the development and analysis of social networks and recommender systems using geometric deep learning techniques and image classification.

ZHIQUAN HE received the M.S. degree from the Institute of Electronics, Chinese Academy of Sciences, in 2001, and the Ph.D. degree from the Department of Computer Science, University of Missouri, Columbia, in 2014. He is currently an Assistant Professor with the College of Information Engineering, Shenzhen University, China. His research interests include image processing, computer vision, and machine learning.

ZHIHAI HE (Fellow, IEEE) was a Research Engineer with the David Sarnoff Research Center. He is currently a Professor with the Electrical Engineering and Computer Science Department, University of Missouri. He was named as a Fellow of the Institute of Electrical and Electronics Engineers (IEEE), in 2015, for his contributions to video communication and visual sensing technologies.