Graph Convolutional Networks: Algorithms, Applications and Open Challenges

Si Zhang[1(✉)], Hanghang Tong[1], Jiejun Xu[2], and Ross Maciejewski[1]

[1] Arizona State University, Tempe, USA
{szhan172,hanghang.tong,rmacieje}@asu.edu
[2] HRL Laboratories, Malibu, USA
jxu@hrl.com

Abstract. Graph-structured data naturally appear in numerous application domains, ranging from social analysis, bioinformatics to computer vision. The unique capability of graphs enables capturing the structural relations among data, and thus allows to harvest more insights compared to analyzing data in isolation. However, graph mining is a challenging task due to the underlying complex and diverse connectivity patterns. A potential solution is to learn the representation of a graph in a low-dimensional Euclidean space via embedding techniques that preserve the graph properties. Although tremendous efforts have been made to address the graph representation learning problem, many of them still suffer from their shallow learning mechanisms. On the other hand, deep learning models on graphs have recently emerged in both machine learning and data mining areas and demonstrated superior performance for various problems. In this survey, we conduct a comprehensive review specifically on the emerging field of graph convolutional networks, which is one of the most prominent graph deep learning models. We first introduce two taxonomies to group the existing works based on the types of convolutions and the areas of applications, then highlight some graph convolutional network models in details. Finally, we present several challenges in this area and discuss potential directions for future research.

Keywords: Graph convolutional networks · Spectral · Spatial

1 Introduction

Graphs naturally arise in many real-world applications, including social analysis [3], fraud detection [1,45], traffic prediction [28], computer vision [31] and many more. By representing the data as graphs, the structural information can be encoded to model the relations among entities, and furnish more promising insights underlying the data. For example, in a transportation network, nodes are often the sensors and edges represent the spatial proximity among sensors. In addition to the temporal information provided by the sensors themselves, the graph structure

X. Chen et al. (Eds.): CSoNet 2018, LNCS 11280, pp. 79–91, 2018.
https://doi.org/10.1007/978-3-030-04648-4_7

modeled by the spatial correlations leads to a prominent improvement in the traffic prediction problem [28]. Moreover, by modeling the transactions among people as a graph, the complex transaction patterns can be mined for synthetic identity detection [45] and money laundering detection [46].

However, the complex structure of graphs [5] often hampers the capability of gaining the true insights underlying the graphs. Such complexity, for example, resides in the non-Euclidean nature of the graph-structured data. A potential solution to deal with the complex patterns is to learn the graph representations in a low-dimensional Euclidean space via embedding techniques, including the traditional graph embedding methods [4,34,37] and the recent network embedding methods [21,33]. Once the low-dimensional representations are learned, many graph-related problems can be easily done, such as the classic node classification and link prediction [21]. There exist many thorough reviews on both traditional graph embedding and recent network embedding methods. For example, [40] reviews several well-established traditional graph embedding methods and discusses the general framework for graph dimensionality reduction. Hamilton et al. review the general graph representation learning methods, including node embedding and subgraph embedding [23]. Furthermore, [11] discusses the differences between the traditional graph embedding and the recent network embedding methods. One notable difference is that the recent network embedding is more suitable for the task-specific network inference. Other existing literature reviews on network embedding include [8,20].

Despite some successes of these embedding methods, many of them suffer from the limitations of the shallow learning mechanisms [21,33] and might fail to discover the more complex patterns behind the graphs. Deep learning models, on the other hand, have been demonstrated their power in many applications. For example, convolution neural networks (CNN) achieve a promising performance in many computer vision [19] and natural language processing [18] applications. In particular, due to the grid-like nature of images, the convolution layers in CNN enable to learn different trainable localized filters which scan every pixel in the images, combining with the surrounding pixels. The basic components are the convolution and pooling operators, as well as the trainable localized filters.

However, the non-Euclidean characteristic of graphs (e.g., the irregular structure) makes the graph convolutions and graph filtering not as well-defined as on images. In the past decades, researchers have been working on the graph signal operations, such as graph filtering, graph wavelets, etc. Shuman et al. give a comprehensive overview of graph signal processing, including the common operations on graphs [36]. To be brief, spectral graph convolutions are defined in the graph Fourier domain, which is considered as an analogy of 1-D signal Fourier transform. Graph filtering can be defined in the spectral and vertex domains. The emergence of these operators open a door to graph convolutional networks. Note that in the past few years, many other graph deep learning models have been proposed, including (but are not limited to): (1) graph auto-encoder [26], (2) graph generative adversarial model [14,44], (3) graph attention model [27,39], (4) graph recurrent neural networks [43]. But in this survey, we focus specifically on reviewing the

existing literature of the graph convolutional networks. The main contributions of this survey are summarized as following:

1. We introduce two taxonomies to group the existing graph convolutional network models by the types of filtering and the areas of applications.
2. We motivate each taxonomy by surveying and discussing the state-of-the-art graph convolutional network models.
3. We discuss the challenges of the current models that need to be addressed and highlight some promising directions for the future work.

The rest of the paper is organized as follows. We start by summarizing the notations and introducing some preliminaries of graph convolutional networks in Sect. 2. Then in Sect. 3 and Sect. 4, we categorize the existing models into the spectral based methods and the spatial based methods by the types of graph filtering with some detailed examples. Section 5 presents the methods from a view of applications. In Sect. 6, we conclude our survey, discuss some of the challenges and provide some directions for the future work.

2 Notations and Preliminary

In this section, we present the notations and some preliminaries for the graph convolutional networks. In general, we use bold uppercase letters for matrices, bold lowercase letters for vectors, and lowercase letters for scalars. For matrix indexing, we use $\mathbf{A}(i,j)$ to denote the entry at the intersection of the i-th row and j-th column. We denote the transpose of a matrix $\mathbf{A}$ as $\mathbf{A}^T$.

Graphs and Graph Signals. In this survey, we are interested in the graph convolutional network models on an undirected connected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$, which consists of a set of nodes $\mathcal{V}$ with $|\mathcal{V}| = n$, a set of edges $\mathcal{E}$ with $|\mathcal{E}| = m$ and the adjacency matrix $\mathbf{A}$. If there is an edge between node i and node j, the entry $\mathbf{A}(i,j)$ denotes the weight of the edge; otherwise, $\mathbf{A}(i,j) = 0$. For unweighted graphs, we simply set $\mathbf{A}(i,j) = 1$. We denote the degree matrix of $\mathbf{A}$ as a diagonal matrix $\mathbf{D}$ where $\mathbf{D}(i,i) = \sum_{j=1}^{n} \mathbf{A}(i,j)$. Then the Laplacian matrix of $\mathbf{A}$ is denoted as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. The corresponding symmetrically normalized Laplacian matrix is $\tilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ where $\mathbf{I}$ is an identity matrix.

A graph signal defined on the nodes is represented as a vector $\mathbf{x} \in \mathbb{R}^n$ where $\mathbf{x}(i)$ is the signal value on the node i [36]. Node attributes, for instance, can be considered as the graph signals. Denote $\mathbf{X} \in \mathbb{R}^{n \times d}$ as the node attribute matrix of an attributed graph, then the columns of $\mathbf{X}$ are the d signals of the graph.

Graph Fourier Transform. It is well-known that the classic Fourier transform of an 1-D signal f is computed by $\hat{f}(\xi) = \langle f, e^{2\pi i \xi t} \rangle$ where ξ is the frequency of $\hat{f}$ in the spectral domain and the complex exponential is the eigenfunction of the Laplace operator. Analogously, the graph Laplacian matrix $\mathbf{L}$ is the Laplace operator defined on a graph, and hence an eigenvector of $\mathbf{L}$ associated with its corresponding eigenvalue is an analog to the complex exponential at a certain frequency. Note that the symmetrically normalized Laplacian matrix $\tilde{\mathbf{L}}$ and the

random-walk transition matrix can be also used as the graph Laplace operator. In particular, denote the eigenvalue decomposition of $\tilde{\mathbf{L}}$ as $\tilde{\mathbf{L}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ where the l-th column of $\mathbf{U}$ is the eigenvector $\mathbf{u}_l$ and $\mathbf{\Lambda}(l,l)$ is the corresponding eigenvalue λ_l, then we can compute the Fourier transform of a graph signal $\mathbf{x}$ as

$$\hat{\mathbf{x}}(\lambda_l) = \langle \mathbf{x}, \mathbf{u}_l \rangle = \sum_{i=1}^{n} \mathbf{x}(i)\mathbf{u}_l^*(i) \tag{1}$$

The above equation represents in the spectral domain a graph signal defined in the vertex domain. Then the inverse graph Fourier transform can be written as

$$\mathbf{x}(i) = \sum_{l=1}^{n} \hat{\mathbf{x}}(\lambda_l)\mathbf{u}_l(i) \tag{2}$$

Graph Filtering. Graph filtering is a localized operation on graph signals. Analogous to the classic signal filtering in the time or spectral domain, one can localize a graph signal in its vertex domain or spectral domain as well.
(1) Frequency filtering: Recall that the frequency filtering of a classic signal is often represented as the convolution with the filter signal in the time domain. However, due to the irregular structure of the graphs (e.g., different nodes having different numbers of neighbors), graph convolution in the vertex domain is not as straightforward as the classic signal convolution in the time domain. Note that for classic signals, the convolution in the time domain is equivalent to the inverse Fourier transform of the multiplication between the spectral representations of two signals. Therefore, the spectral graph convolution is defined analogously as

$$(\mathbf{x} *_{\mathcal{G}} \mathbf{y})(i) = \sum_{l=1}^{n} \hat{\mathbf{x}}(\lambda_l)\hat{\mathbf{y}}(\lambda_l)\mathbf{u}_l(i) \tag{3}$$

Note that $\hat{\mathbf{x}}(\lambda_l)\hat{\mathbf{y}}(\lambda_l)$ indicates the filtering in the spectral domain. Thus, the frequency filtering of a signal $\mathbf{x}$ on graph $\mathcal{G}$ with a filter $\mathbf{y}$ is exactly same as Eq. (3) and is further re-written as

$$\mathbf{x}_{out} = \mathbf{x} *_{\mathcal{G}} \mathbf{y} = \mathbf{U} \begin{bmatrix} \hat{\mathbf{y}}(\lambda_1) & & 0 \\ & \ddots & \\ 0 & & \hat{\mathbf{y}}(\lambda_n) \end{bmatrix} \mathbf{U}^T\mathbf{x} \tag{4}$$

(2) Vertex filtering: The graph filtering of a signal $\mathbf{x}$ in the vertex domain is generally defined as a linear combination of the signal components in the nodes neighborhood. Mathematically, the vertex filtering of a signal $\mathbf{x}$ at node i is

$$\mathbf{x}_{out}(i) = w_{i,i}\mathbf{x}(i) + \sum_{j \in \mathcal{N}(i,K)} w_{i,j}\mathbf{x}(j) \tag{5}$$

where $\mathcal{N}(i,K)$ represents the K-hop neighborhood of node i in the graph and the parameters $\{w_{i,j}\}$ are the weights used for the combination. It can be shown that by using a K-polynomial filter, the frequency filtering can be interpreted from the vertex filtering perspective [36].

3 Spectral Graph Convolutional Networks

In this section and the subsequent Sect. 4, we categorize the graph convolutional neural networks into the spectral based methods and the spatial based methods respectively. We consider the spectral based methods to be those methods that start with constructing the frequency filtering.

The first notable spectral based graph convolutional network is proposed by Bruna et al. [7]. Motivated by the classic CNN, this deep model on graphs contains several spectral convolutional layers that take a vector $\mathbf{X}_p$ of size $n \times d_p$ as the input feature map and output a feature map $\mathbf{X}_{p+1}$ of size $n \times d_{p+1}$ by:

$$\mathbf{X}_{p+1}(:,j) = \sigma\left(\sum_{i=1}^{d_p} \mathbf{V} \begin{bmatrix} (\boldsymbol{\theta}_i^j)(1) & & 0 \\ & \ddots & \\ 0 & & (\boldsymbol{\theta}_i^j)(n) \end{bmatrix} \mathbf{V}^T \mathbf{X}_p(:,i)\right), \ \forall j = 1, \cdots, d_{p+1} \tag{6}$$

where $\mathbf{X}_p(:,i)$ ($\mathbf{X}_{p+1}(:,j)$) is the i-th (j-th) dimension of the input (output) feature map respectively, $\boldsymbol{\theta}_i^j$ denotes a vector of learnable parameters of the filter $\boldsymbol{\theta}_i^j$. Each column of $\mathbf{V}$ is the eigenvector of $\mathbf{L}$ and $\sigma(\cdot)$ is the activation function. However, there are several issues with this convolutional structure. First, the eigenvector matrix $\mathbf{V}$ requires the explicit computation of the eigenvalue decomposition of the graph Laplacian matrix, and hence suffers from the $O(n^3)$ time complexity which is impractical for large-scale graphs. Second, though the eigenvectors can be pre-computed, the time complexity of Eq. (6) is still $O(n^2)$. Third, there are $O(n)$ parameters to be learned in each layer. Besides, these non-parametric filters are not localized in the vertex domain. To overcome the limitations, the authors also propose to use a rank-r approximation of eigenvalue decomposition. To be specific, they use the first r eigenvectors of $\mathbf{V}$ that carry the most smooth geometry of the graph and consequently reduce the number of parameters of each filter to $O(1)$ [7]. Moreover, if the graph contains the clustering structure that can be explored via such a rank-r factorization, the filters are potentially localized. However, it still requires $O(n^2)$ time complexity.

To address these limitations, Defferrard et al. propose to use K-polynomial filters in the convolutional layers for localization [12]. Such a K-polynomial filter is represented by $\hat{\mathbf{y}}(\lambda_l) = \sum_{k=1}^{K} \theta_k \lambda_l^k$. As mentioned in Sect. 2, the K-polynomial filters achieve a good localization by integrating the node features within the K hop neighborhood [36], and the number of the trainable parameters decreases to $O(K) = O(1)$. In addition, to further reduce the computational complexity, the Chebyshev polynomial approximation [24] is used to compute the spectral graph convolution. Mathematically, the Chebyshev polynomial $T_k(x)$ of order k can be recursively computed by $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$, $T_1(x) = x$. They normalize the filters by $\tilde{\lambda}_l = 2\frac{\lambda_l}{\lambda_{\max}} - 1$ to make the scaled eigenvalues lie within $[-1, 1]$ [12]. As a result, the convolution layer is

$$\mathbf{X}_{p+1}(:,j) = \sigma\left(\sum_{i=1}^{d_p} \sum_{k=0}^{K-1} (\boldsymbol{\theta}_i^j)(k+1) T_k(\tilde{\mathbf{L}}) \mathbf{X}_p(:,i)\right), \ \forall j = 1, \cdots, d_{p+1} \tag{7}$$

where $\boldsymbol{\theta}_i^j$ is a K-dimensional parameter vector for the i-th column of input feature map and the j-th column of output feature map. The authors also design a max pooling operation [12] with the multilevel clustering method Graclus [13] which is quite efficient to uncover the hierarchical structure of the graphs.

As a special variant, the graph convolutional network proposed by Kipf et al. (named as GCN) aims at the semi-supervised node classification task on graphs [25]. In this model, the authors truncate the Chebyshev polynomial to first-order (i.e., $K = 2$ in Eq. (7)) and specifically set $(\boldsymbol{\theta})_i^j(1) = -(\boldsymbol{\theta})_i^j(2) = \theta_i^j$. Besides, since the eigenvalues of $\tilde{\mathbf{L}}$ are within $[0, 2]$, relaxing $\lambda_{\max} = 2$ still guarantees $-1 \leq \tilde{\lambda}_l \leq 1,\ \forall l = 1, \cdots, n$. This leads to the simplified convolution layer as

$$\mathbf{X}_{p+1} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}_p\boldsymbol{\Theta}_p\right) \tag{8}$$

where $\tilde{\mathbf{A}} = \mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ and $\tilde{\mathbf{D}}$ is the diagonal degree matrix of $\tilde{\mathbf{A}}$, $\boldsymbol{\Theta}_p$ is a $d_{p+1} \times d_p$ parameter matrix. Besides, Eq. (8) has a close relationship with the Weisfeiler-Lehman isomorphism test [35]. The last layer outputs the node representations. A softmax classifier is then added after the last spectral convolutional layer and the objective is to minimize the cross-entropy error over the labeled nodes. The objective function is then minimized in a gradient descent manner. However, the training process could be costly (in terms of memory) for large-scale graphs. Moreover, the transduction of GCN interferes with the generalization, making the learning of representations of the unseen nodes in the same graph and the nodes in an entirely different graph more difficult [25].

To address the issues of GCN [25], FastGCN [10] improves the original GCN model by viewing the spectral graph convolution as an integral of embedding functions under some probability measure. It first assumes the input graph $\mathcal{G}$ is an induced subgraph of a possibly infinite graph $\mathcal{G}'$ such that the nodes $\mathcal{V}$ of $\mathcal{G}$ are i.i.d. samples of the nodes of $\mathcal{G}'$ (denoted as $\mathcal{V}'$) under some probability measure $\mathcal{P}$. This way, the original convolution layer represented by Eq. (8) can be illustrated by an embedding function of independent vertices. Denote the embedding function at the p-th layer as $\mathbf{x}_p$, then we have

$$\mathbf{x}_{p+1}(v) = \sigma\left(\int \tilde{\mathbf{A}}(v, u)\mathbf{x}_p(u)\boldsymbol{\Theta}_p d\mathcal{P}(u)\right) \tag{9}$$

where u, v are some independent nodes. Now, Eq. (9) can be approximated by Monte Carlo sampling. Denote some i.i.d. samples $u_1^p, \cdots, u_{t_p}^p$ at layer-p, the integral can be estimated by

$$\mathbf{x}_{p+1}(v) = \sigma\left(\frac{1}{t_p}\sum_{i=1}^{t_p}\tilde{\mathbf{A}}(v, u_i^p)\mathbf{x}_p(u_i^p)\boldsymbol{\Theta}_p\right) \tag{10}$$

Denote P as the number of layers of the deep architecture, and $u_1^P, \cdots, u_{t_P}^P$ as a batch of nodes. At each layer p, they uniformly sample with replacement the

nodes $u_1^p, \cdots, u_{t_p}^p$, then the output feature map is computed by

$$\mathbf{X}_{p+1}(v,:) = \sigma\left(\frac{n}{t_p}\sum_{i=1}^{t_p}\tilde{\mathbf{A}}(v,u_i^p)\mathbf{X}_p(u_i^p,:)\mathbf{\Theta}_p\right) \tag{11}$$

and the batch loss w.r.t. the output of the last layer is

$$\mathcal{L} = \frac{1}{t_P}\sum_{i=1}^{t_P} g\left(\mathbf{X}_P(u_i^P,:)\right) \tag{12}$$

where $g(\cdot)$ is some loss function. Note that this Monte Carlo estimator of the original convolution could lead to a high variance of estimation. To reduce the variance, the authors also formalize the variance and solve for a sampling distribution $\mathcal{P}$ of nodes. Due to the space limitation, we suggest the readers of interests to refer to [10]. In addition, [9] is another recent work on the stochastic training of GCN [25]. To reduce the variance of the estimator, the authors use the historical activations of nodes as a control variate and propose an efficient sampling-based stochastic algorithm. Besides, the authors theoretically prove the convergence of the algorithm regardless of the sampling size in the training phase, and also the exact predictions in the testing phase in [9].

4 Spatial Graph Convolutional Networks

As the spectral graph convolution relies on the specific eigenfunctions of Laplacian matrix, it is nontrivial to transfer the spectral based graph convolutional network models learned on one graph to another graph whose eigenfunctions are different. Spatial based methods, on the other hand, alternatively generalize the convolution to the combinations of the graph signal within the nodes neighborhood and define the learnable filters in the vertex domain.

Monti et al. propose a generic graph convolution network framework named MoNet [31] by designing a universe patch operator which integrates the signals within the node neighborhood. In particular, for a node i and its neighboring node $j \in \mathcal{N}(i)$, they define a d-dimensional pseudo-coordinates $\mathbf{u}(i,j)$ and feed it into P learnable kernel functions $(w_1(\mathbf{u}), \cdots, w_P(\mathbf{u}))$. Then the patch operator is formulated as $D_p(i) = \sum_{j\in\mathcal{N}(i)} w_p(\mathbf{u}(i,j))\mathbf{x}(j),\ p = 1, \cdots, P$ where $\mathbf{x}(j)$ is the signal value at the node j. The graph convolution in the spatial domain is then based on the patch operator as

$$(\mathbf{x} *_s \mathbf{y})(i) = \sum_{l=1}^{P} \mathbf{g}(p) D_p(i)\mathbf{x} \tag{13}$$

It is shown that by carefully selection of $\mathbf{u}(i,j)$ and the kernel function $w_p(\mathbf{u})$, many existing graph convolutional network models [2,25] can be viewed as a specific case of MoNet. SplineCNN [15] follows the same framework (i.e., Eq. (13)) but uses a different convolution kernel based on B-splines.

From a more general perspective, the graph convolution in the spatial domain can be alternatively thought of as an aggregation of a subset of nodes. Hamilton et al. propose an aggregation based representation learning, named GraphSAGE [22]. The full batch version of the algorithm is straightforward: for a node i, one (1) aggregates the representation vectors of all its immediate neighbors in the current layer via some learnable aggregator; (2) concatenates the representation vector of node i with the aggregated representation; (3) then feeds the concatenated vector to a fully connected layer with some nonlinear activation function $\sigma(\cdot)$, followed by a normalization step. The output of the last layer is considered as the final representations of nodes, which can be followed by some loss function. The authors provide some choices of the aggregator functions, including the mean aggregator, LSTM aggregator and the pooling aggregator. Among others, using the mean aggregator makes the whole algorithm approximately resemble the GCN model [25]. In addition, for training efficiency, they also provide a minibatch variant by uniformly sampling the neighboring nodes [22].

Velickovic et al. design a novel attention layer that aggregates the features of the neighboring nodes weighted by some learnable importance [39]. Consider the input node attribute matrix $\mathbf{X}$ with each row as the feature vector of a node. The attention layer contains a shared learnable weight matrix $\mathbf{W}$ and computes the attention coefficients between node i and its neighbor node $j \in \mathcal{N}(i)$ by

$$\alpha_{ij} = \frac{\exp\left(\mathbf{a}^T[\mathbf{W}\mathbf{X}(i,:)^T \| \mathbf{W}\mathbf{X}(j,:)^T]\right)}{\sum_{q\in\mathcal{N}(i)} \exp\left(\mathbf{a}^T[\mathbf{W}\mathbf{X}(i,:)^T \| \mathbf{W}\mathbf{X}(q,:)^T]\right)} \tag{14}$$

where $\|$ denotes the concatenation operation and $\mathbf{a}$ is a single-layer feedforward neural network. This attention coefficient acts as a weight to encode the importance of feature vector of the neighboring node j for node i. And the final output of the feature vector is computed by a linear combination $\mathbf{X}_{out}(i,:) = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}\mathbf{W}\mathbf{X}(j,:)^T\right)$. To stabilize the learning process, the authors apply the multi-head attention [38] (i.e., L independent attention mechanism as Eq. (14)), and then feed the average of the output of all heads to a nonlinearity. Compared to the GCN model [25], more flexibility is achieved thanks to the learnable importance of the nodes within the neighborhood.

Note that despite the inherent differences among the models above, all of them can be viewed as an instance of using vertex filtering. It is just the strategy of how to decide the weights w_{ij} in Eq. (5) that differentiates the models.

5 Applications of Graph Convolutional Networks

The different graph convolutional network models can be also divided by what kind of data they are applied to. Although a substantial amount of applications exist, we generally categorize them into (1) applications on graph data, (2) applications on image and manifold, and (3) applications on other data.

Applications on Graph Data. A number of works have been proposed to solve the tasks on graphs. The majority of them are for node classification, including [10,17,22,25,31,39]. A commonality among them is that the output feature

map of these methods can be considered as the node representations, and thus these methods can be also naturally generalized to other node-level problems, such as link prediction, node clustering and visualization. Another application is the graph classification. One straightforward way is to aggregate the learned node representations as the graph representations and then feed to some classifiers (e.g., fully connected network). However, this may not be a quite promising strategy since the simple aggregation of the isolated node representations may not represent the graph in its entirety. [7,12,42] leverages the graph coarsening and pooling operator to explore the hierarchical representations of graphs. In particular, [42] recently designs a differential pooling operator that can generate the graph hierarchical representations. There are some other adapted graph convolutional network models that aim to solve problems in specific domains. For example, Li et al. [28] propose a diffusion convolutional recurrent neural network for traffic forecasting by exploring spatial and temporal dependencies. [16] introduces a special graph convolutional network architecture for protein interface prediction.

Applications on Images and Manifolds. Image classification problems have been studied for decades. Traditional CNN based methods directly consider the images as a grid-like structure. The recent graph convolutional network models allow to consider image classification as a classification on the non-Euclidean structures (e.g., graphs that encode the relations among pixels). Briefly speaking, k-NN similarity graphs with pixels of the images as the nodes need to be constructed and the image classification problem is then converted to a graph classification problem. Existing works on this problem include [7,12,31], etc. In addition, another application of the graph convolutional network models in the computer vision area is to learn the correspondence between the collections of 3D shapes represented by the discrete manifolds. This problem is roughly cast as a labelling problem, i.e., to label each node on a query shape with the index of the node on the target shape [31].

Applications on Other Data. In addition to the applications on graphs and manifolds, graph convolutional network models are also widely used for natural language processing. For example, [30] deals with the semantic role labelling by encoding sentences with the graph convolutional network. Marcheggiani et al. attempt to use graph convolutional network models for machine translation problems [29]. Besides, they can also be used for recommender systems. In particular, Monti et al. cast the recommender system problem as a matrix completion problem with two graphs as side information, then define a multiple graph convolution operator of the convolution layer to adapt the graph convolutional network model to solve the matrix completion problem [32]. Another notable work [41] deploys a random-walk-based graph convolutional network model for high-quality recommendations. Besides, the authors develop an on-the-fly convolution computation for efficient training process and a MapReduce pipeline for efficient inferences.

6 Concluding Remarks

Graph convolutional network models, as one category of the graph deep learning (or geometric deep learning) models, have become a very hot topic in both machine learning and data mining areas, and a substantial amount of models have been proposed to solve different problems. In this survey, we conduct a comprehensive literature review on the emerging field of graph convolutional networks. Specifically, we introduce two intuitive taxonomies to group the existing works. These are based on the types of graph filtering operations, and based on the areas of applications. For each taxomony, we highlight with some detailed examples from a unique standpoint. In addition to our survey, another comprehensive tutorial on geometric deep learning [6] may help readers step into this area Meanwhile, despite the advancements made by the recent works, there still exist some potential issues in the current graph convolutional network models. This way we discuss some challenges and provide some potential future directions.

Multiple Graph Convolutional Networks. As already mentioned before, the major drawback of the spectral graph convolutional networks is its inability of adaptation from one graph to another graph if two graphs have different Fourier basis (i.e., eigenfunctions of the Laplacian matrix). The existing work [32] alternatively learns the filter parameters by generalizing the eigenfunctions of a single graph to the eigenfunctions of the Kronecker product graph of multiple input graphs. As a different track, the spatial graph convolutional network models attempt to learn the a rule of how to combine neighboring nodes in the vertex domain which could be used on different graphs. However, a drawback of these methods is the inability of modeling the interactions (e.g., anchor links) or correlations (e.g., correlations among multiple views) across multiple graphs. In fact, given multiple graphs, the representation learning of a unique node should be able to benefit from more information provided across graphs or views. However, to our best knowledge, there is no existing model aiming at the problems in this setting.

Hybrid Spectral-Spatial Graph Convolutional Networks. Note that the graph convolutional network models reviewed in this survey start with either the spectral filtering in the frequency domain or the spatial filtering in the vertex domain. This raises the issue that the existing graph convolutional network models may not fully exploit the insights simultaneously from both the spectral and spatial perspectives of the graph. Recall that the anomaly detection on some classic 1-D signals requires the knowledge in both time domain and frequency domain. In this way, a hybrid spectral-spatial graph convolution operator may provide more comprehensive representations of nodes and hence help some tasks, such as anomaly detection on graphs.

Deep Graph Convolutional Networks. Although the initial objective of graph convolutional network models is to leverage the deep architecture for better representation learning, most of the current models still suffer from their shallow structure. For example, GCN [25] in practice only uses two layers. And as the authors analyzed, more convolution layers may even hurt the performance [25].

This is also intuitive due to its simple propagation procedure. As deeper the architecture is, the representations of nodes may become smoother even for those nodes that are distinct and far from each other. This issue violates the purpose of using deep models. Consequently, how to build a deep architecture that exploits the deeper structural patterns of graphs is another possible research direction.

Acknowledgement. This material is supported by the National Science Foundation under Grant No. IIS-1651203, IIS-1715385, IIS-1743040, and CNS-1629888, by DTRA under the grant number HDTRA1-16-0017, by the United States Air Force and DARPA under contract number FA8750-17-C-0153 (Distribution Statement "A" (Approved for Public Release, Distribution Unlimited)), by Army Research Office under the contract number W911NF-16-1-0168, and by the U.S. Department of Homeland Security under Grant Award Number 2017-ST-061-QA0001. The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

1. Akoglu, L., Tong, H., Koutra, D.: Graph based anomaly detection and description: a survey. Data Min. Knowl. Disc. **29**(3), 626–688 (2015)
2. Atwood, J., Towsley, D.: Diffusion-convolutional neural networks. In: NIPS (2016)
3. Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: WSDM, pp. 635–644. ACM (2011)
4. Belkin, M., Niyogi, P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In: NIPS, pp. 585–591 (2002)
5. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.U.: Complex networks: structure and dynamics. Phys. Rep. **424**(4–5), 175–308 (2006)
6. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Process. Mag. **34**(4), 18–42 (2017)
7. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
8. Cai, H., Zheng, V.W., Chang, K.: A comprehensive survey of graph embedding: problems, techniques and applications. TKDE (2018)
9. Chen, J., Zhu, J., Song, L.: Stochastic training of graph convolutional networks with variance reduction. In: ICML, pp. 941–949 (2018)
10. Chen, J., Ma, T., Xiao, C.: FastGCN: fast learning with graph convolutional networks via importance sampling. arXiv preprint arXiv:1801.10247 (2018)
11. Cui, P., Wang, X., Pei, J., Zhu, W.: A survey on network embedding. TKDE (2018)
12. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS, pp. 3844–3852 (2016)
13. Dhillon, I.S., Guan, Y., Kulis, B.: Weighted graph cuts without eigenvectors a multilevel approach. IEEE Trans. Pattern Anal. Mach. Intell. **29**(11) (2007)
14. Ding, M., Tang, J., Zhang, J.: Semi-supervised learning on graphs with generative adversarial nets. arXiv preprint arXiv:1809.00130 (2018)
15. Fey, M., Lenssen, J.E., Weichert, F., Müller, H.: SplineCNN: fast geometric deep learning with continuous b-spline kernels. In: CVPR, pp. 869–877 (2018)

16. Fout, A., Byrd, J., Shariat, B., Ben-Hur, A.: Protein interface prediction using graph convolutional networks. In: NIPS, pp. 6530–6539 (2017)
17. Gao, H., Wang, Z., Ji, S.: Large-scale learnable graph convolutional networks. In: KDD, pp. 1416–1424. ACM (2018)
18. Gehring, J., Auli, M., Grangier, D., Dauphin, Y.N.: A convolutional encoder model for neural machine translation. arXiv preprint arXiv:1611.02344 (2016)
19. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR, pp. 580–587 (2014)
20. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: a survey. Knowl. Based Syst. **151**, 78–94 (2018)
21. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: KDD, pp. 855–864. ACM (2016)
22. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS, pp. 1024–1034 (2017)
23. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: methods and applications. arXiv preprint arXiv:1709.05584 (2017)
24. Hammond, D.K., Vandergheynst, P., Gribonval, R.: Wavelets on graphs via spectral graph theory. Appl. Comput. Harmonic Anal. **30**(2), 129–150 (2011)
25. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
26. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
27. Lee, J.B., Rossi, R., Kong, X.: Graph classification using structural attention. In: KDD, pp. 1666–1674. ACM (2018)
28. Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: data-driven traffic forecasting (2018)
29. Marcheggiani, D., Bastings, J., Titov, I.: Exploiting semantics in neural machine translation with graph convolutional networks. arXiv preprint arXiv:1804.08313 (2018)
30. Marcheggiani, D., Titov, I.: Encoding sentences with graph convolutional networks for semantic role labeling. arXiv preprint arXiv:1703.04826 (2017)
31. Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model CNNs. In: CVPR, vol. 1, p. 3 (2017)
32. Monti, F., Bronstein, M., Bresson, X.: Geometric matrix completion with recurrent multi-graph neural networks. In: NIPS, pp. 3697–3707 (2017)
33. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: KDD, pp. 701–710. ACM (2014)
34. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**(5500), 2323–2326 (2000)
35. Shervashidze, N., Schweitzer, P., Leeuwen, E.J.V., Mehlhorn, K., Borgwardt, K.M.: Weisfeiler-lehman graph kernels. JMLR **12**(Sep), 2539–2561 (2011)
36. Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., Vandergheynst, P.: The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Process. Mag. **30**(3), 83–98 (2013)
37. Tenenbaum, J.B., De Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
38. Vaswani, A., et al.: Attention is all you need. In: NIPS, pp. 5998–6008 (2017)
39. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)

40. Yan, S., Xu, D., Zhang, B., Zhang, H.J., Yang, Q., Lin, S.: Graph embedding and extensions: a general framework for dimensionality reduction. IEEE Trans. Pattern Anal. Mach. Intell. **29**(1), 40–51 (2007)
41. Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. arXiv preprint arXiv:1806.01973 (2018)
42. Ying, R., You, J., Morris, C., Ren, X., Hamilton, W.L., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. arXiv preprint arXiv:1806.08804 (2018)
43. You, J., Ying, R., Ren, X., Hamilton, W.L., Leskovec, J.: GraphRNN: a deep generative model for graphs. arXiv preprint arXiv:1802.08773 (2018)
44. Yu, W., et al.: Learning deep network representations with adversarially regularized autoencoders. In: KDD, pp. 2663–2671. ACM (2018)
45. Zhang, S., et al.: Hidden: hierarchical dense subgraph detection with application to financial fraud detection. In: SDM, pp. 570–578. SIAM (2017)
46. Zhou, D., et al.: A local algorithm for structure-preserving graph cut. In: KDD, pp. 655–664. ACM (2017)