Scalable Graph Convolutional Networks With Fast Localized Spectral Filter for Directed Graphs

CHENSHENG LI, XIAOWEI QIN, XIAODONG XU, DUJIA YANG, AND GUO WEI
CAS Key Laboratory of Wireless-Optical Communications, University of Science and Technology of China, Hefei 230026, China
Corresponding author: Xiaowei Qin (qinxw@ustc.edu.cn)

This work was supported by the National Key Research and Development Program of China under Grant 2018YFA0701603.

ABSTRACT Graph convolutional neural networks (GCNNs) have been emerged to handle graph-structured data in recent years. Most existing GCNNs are either spatial approaches working on neighborhood of each node, or spectral approaches based on graph Laplacian. Compared with spatial-based GCNNs, spectral-based GCNNs are capable of highly exploiting graph structure information, but always regard graphs undirected. Actually, there are many scenarios where the graph structures are directed, such as social networks, citation networks, etc. Treating graphs undirected may lose important information, which is helpful for graph learning tasks. This motivate us to construct a spectral-based GCNN for directed graphs. In this paper, we propose a scalable graph convolutional neural network with fast localized convolution operators derived from directed graph Laplacian, which is called fast directed graph convolutional network (FDGCN). FDGCN can directly work on directed graphs and can scale to large graphs as the convolution operation is linear with the number of edges. Furthermore, we find that FDGCN can unify the graph convolutional network (GCN), which is a classic spectral-based GCNN. The mechanism of FDGCN is thoroughly analyzed from spatial aggregation point of view. Since previous work has confirmed that considering uncertainty of graph could promote GCN a lot, the proposed FDGCN is further enhanced through extra training epochs on random graphs generated by mixed membership stochastic block model (MMSBM). Experiments are conducted for semi-supervised node classification tasks to evaluate the performance of FDGCN. Results show that our model can outperform or match state-of-the-art models in most cases.

INDEX TERMS Graph convolutional networks, directed graph Laplacian, spectral graph theory, semi-supervised learning.

I. INTRODUCTION

Deep learning models like convolutional neural networks (CNNs) have shown great success in tackling learning problems such as image classification [1], object detection [2], image inpainting [3], etc. However, these CNN models are mostly developed for grid-like structured data, and cannot be directly applied for graph-structured data. In fact, graph-structured data is common in real world, such as social networks, 3D meshes in graphics, molecules of materials and drugs, etc. Therefore, there has been a growing interest in constructing deep neural models on graph-structured data which are called graph neural networks (GNNs) [4], [5]. The idea of GNN models can be traced in early works, which rely on processing and propagating information recursively across the graph [6]. Unfortunately, the unsatisfactory properties of convergence and scalability make them hard to use. To combat this dilemma, a significant body of works have been emerged to construct convolution neural networks on graphs, i.e., graph convolutional neural networks (GCNNs).

GCNN models can be categorized into two branches, according to the means of constructing convolution operator (i.e., filter). One is spatial-based GCNN models, where the key is appropriately aggregating information of neighborhood of each node. In early researches, recurrent units are used to aggregate information of all neighbor nodes [7]–[9]. Subsequently, some attempts are made to apply standard CNN on fixed size of neighborhood for each node [10], [11]. Recently, different new mechanisms have been merged into graph learning. Attention mechanism is used in [12]–[14] to learn weight ratios for each neighbor node, while authors in [15] use differentiable pooling
strategy. Adaptive receptive paths is considered in [16] and cluster technique is used in [17]. Another branch is spectral-based GCNN models, which uses spectral filters based on graph Laplacian to construct network models. The first proposals define spectral filters on graphs by using the first few eigenvectors of graph Laplacian [9], [18], [19]. ChebyNet proposed in [20] uses Chebyshev polynomials to approximate the spectral filter. Thus, it first achieves a spectral-free filter, which can greatly reduce the complexity as there is no need to do the eigen-decomposition. Kipf and Welling in [21] take a further step by fixing Chebyshev polynomials to 1st-order. The model, called graph convolutional network (GCN), is well-known due to its impressives results. The following works either try to improve GCN with different strategies [22]–[24], or construct a graph convolutional model with different mathematical tools [25], [26]. Authors in [27]–[29] extend GCN model to hypergraph scenario in different ways, which make GCN model more complete. Works [30] and [31] try to generalize existing graph convolution networks into a unified framework. The former uses a message passing mechanism and the latter uses pseudo-coordinates one to give the explanation.

Those GCNN models all have their own applicable scenarios and limitations. Although Spatial-based models can be generalized to new graphs as they perform convolution locally on each node, they cannot fully integrate graph structure information. Moreover, much free parameters are required to be learned due to aggregating information of neighborhood, which mainly determines the effect of spatial methods. On the contrary, spectral-based models can alleviate the aforementioned problems occurred in spatial-based models. However, there exist two main disadvantages for its application. First, spectral-based models are usually computation unfriendly because of the need to perform eigen-decomposition. To cope with the problem, Chebyshev polynomials is adopted to approximate the spectral filter in ChebyNet [20] and GCN [21]. Second, as far as we know, most previous spectral methods treat graphs undirected, which would lose important direction information as there are lots of scenarios in which edges are directed. It is in demand to construct a convolution operator on directed graphs. Fortunately, Fan Chung gives the definition of directed graph Laplacian operator [32], which provides us a tool to construct the directed graph convolution operator. However, matrix decomposition is needed to evaluate the Perron vector during computing Laplacian. Initial attempts have been taken to construct spectral filter with directed graph Laplacian, where the corresponding model is named DGCN [33]. Nevertheless, it cannot effectively handle large-size graphs due to the time-consuming matrix decomposition in constructing filter with directed graph Laplacian.

To overcome the above challenges, in this paper, we propose a scalable graph convolutional network named fast directed graph convolutional network (FDGCN) for directed graphs with fast localized spectral filters (i.e., convolution operators). The main contributions of our work are summarized as follows:

- We design a novel fast localized convolution operator which could directly work on directed graphs. The directed graph convolution operator is firstly derived by 1st-order Chebyshev polynomials approximation for spectral filter of directed graph Laplacian, which makes FDGCN 1-hop localized. Then the derived convolution operator is further approximated by fixing the Perron vector in the definition of directed graph Laplacian. By doing so, the computational complexity of convolution operation is reduced to be linear with the number of edges, which makes FDGCN can scale to large-size graphs. To our best knowledge, this is the first fast localized convolution operator for directed graphs based on directed graph Laplacian.

- We design a specific FDGCN model for directed graphs with two layers, each of which consists of two different directed graph convolution operators. It is found that FDGCN can generalize GCN as a particular instance thereof. Further we promote it by integrating a graph generation method, i.e., mixed membership stochastic block model (MMSBM) [34].

- We put forward a spatial aggregation point of view to take deep insight into FDGCN. It gives explanations for key derivations of the model. Moreover, it is shown that such point of view can help to better understand some existing models.

- Experiments are conducted for semi-supervised node classification tasks to evaluate the performance of FDGCN. The results show that FDGCN can outperform or match other state-of-the-art models in most cases.

The remainder of this paper is structured as follows. Preliminaries including GCN and directed graph Laplacian are elaborated in section II. In section III, we propose the FDGCN model and give detailed derivations. Then it is figured out that FDGCN can unify GCN. In section IV, mechanisms of FDGCN, together with several other models, are analyzed from spatial aggregation point of view. In section V, extensive experiments are conducted on three datasets with five kinds of models including two implementations of FDGCN, following with comprehensive analysis. In section VI, we make summary of our work and propose potential future directions.

II. PRELIMINARIES

Consider a graph \( G = (\mathcal{V}, \mathcal{E}) \) with node set \( \mathcal{V} \) and edge set \( \mathcal{E} \). Denote the nonnegative adjacent matrix by \( A = [a_{ij}] \in \mathbb{R}^{n \times n} \), where \( n \) is the size of \( \mathcal{V} \). The corresponding degree matrix is denoted by \( D = \text{diag}(d_1, d_2, \ldots, d_n) \), where \( d_i = \sum a_{ij} \) represents for the degree of node \( i \).

A. SPECTRAL GRAPH CONVOLUTIONS

Spectral graph convolutions operate filters on signal \( x \in \mathbb{R}^n \) in the graph Fourier domain. Denote a Fourier-domain filter
by \( f_\theta = \text{diag}(\theta) \), where \( \theta \) represents for the filter parameters. Then the filtering process can be formulated as:

\[
f_\theta * x = F^{-1} f_\theta F x,
\]

where \( F \) is the graph Fourier transform matrix and \( F^{-1} \) represents for the Fourier inverse transform matrix.

**B. SPECTRAL-BASED GCNNs**

As for undirected graphs, its normalized symmetric Laplacian is defined as \( L_{\text{sym}} := D^{-1/2}L D^{-1/2} \) [35], [36]. Almost all spectral-based GCNNs use \( L_{\text{sym}} \) to construct filters. As \( L_{\text{sym}} \) is real symmetric and semi-definite, it can be diagonalized as \( L_{\text{sym}} = I_n - D^{-1/2}AD^{-1/2} = U \Lambda U^T \), where \( \Lambda \) is the diagonal eigenvalue matrix and \( U^T \) corresponds to the Fourier transform matrix. Then we can rethink a filter \( f_\theta \) as a function of \( \Lambda \), i.e., \( f_\theta(\Lambda) \). By approximating the filter with an expansion of \( K \)-th order Chebyshev polynomials [37], we can further get:

\[
f_\theta * x \approx \sum_{k=0}^{K} \theta_k T_k(\hat{\Lambda}) U^T x
\]

where the two rescaled operators, i.e., \( \hat{A} = \frac{2}{\lambda_{\text{max}}} A - I_n \) and \( \hat{L} = \frac{2}{\lambda_{\text{max}}} L_{\text{sym}} - I_n \), map eigenvalues from \([0, \lambda_{\text{max}}]\) to \([-1, 1]\). The above equation can be easily derived as \((U \Lambda U^T)^k = U \Lambda^k U^T\). The Chebyshev polynomials satisfy \( T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \) when \( k \geq 2 \) and \( T_0(x) = 1 \), \( T_1(x) = x \), otherwise. With this expansion of \( K \)-th order Chebyshev polynomials in Laplacian, a spectral graph filter can be performed without Fourier transform, which implies no eigen-decomposition is required. Therefore, the computational complexity of the entire convolution operation on signal \( x \) is greatly reduced from \( \mathcal{O}(n^2) \) to \( \mathcal{O}(K|E|) \), as \( L_{\text{sym}} \) is usually sparse.

ChebNet in [20] exploits such \( K \)-th order Chebyshev polynomials to construct convolutional layers on graphs. GCN proposed in [21] takes a further step by fixing \( K = 1 \) and \( \lambda_{\text{max}} = 2 \). After constraining the relation of the two free parameters and applying a renormalization trick, the final convolution becomes:

\[
f_\theta * x = \theta \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} x,
\]

where \( \tilde{A} = A + I_n \) and \( \tilde{D}_{ii} = \sum_j \tilde{A}_{ij} \). Although the convolution of GCN has a simple form, it works much better than previous models. The authors in [23] attempt to explain why GCN could work so well. They figure out that the achieved benifit is due to the fact, that the graph convolution is a specific case of the Laplacian smoothing [38].

**C. THE LAPLACIAN FOR A DIRECTED GRAPH**

The normalized graph Laplacian \( L_{\text{sym}} \) has shown great power in spectral graph theory. Other than the adjacent matrix \( A \), the eigenvalues of \( L_{\text{sym}} \) are closely related to many major graph invariants [35], which gives the ability for spectral graph theory to deduce the structure and principal properties of a graph. Nevertheless, \( L_{\text{sym}} \) is defined on undirected graph. In the directed graph scenario, the use of \( L_{\text{sym}} \) would cause information of edges to be lost, which brings us great interest in defining Laplacian for a directed graph.

Fortunately, Fan Chung gives definition of the desired directed graph Laplacian operator and examines characteristics of the corresponding eigenvalues with the help of Rayleigh quotient in [32]. This directed graph Laplacian provides us a powerful tool to handle problems in directed graphs.

1) TRANSITION PROBABILITY MATRIX

For a directed graph, it is required to first define the corresponding transition probability matrix \( P \), where the element \( P(u, v) \) denotes the probability of transfer from node \( u \) to node \( v \). Given a directed graph \( G = (V, E) \), the definition of transition probability matrix is:

\[
P(u, v) = \begin{cases} 
\frac{1}{d_u} & \text{if } A(u, v) \neq 0, \\
0 & \text{otherwise}. 
\end{cases}
\]

2) PERRON VECTOR

The definition of Perron vector comes from the famous Perron-Frobenius Theorem. The Perron-Frobenius Theorem indicates that an irreducible nonnegative matrix \( M \) has a unique positive eigenvector \( u \), and the corresponding positive eigenvalue \( \rho \) is the upper bound of the absolute values of all eigenvalues [39]. For a transition probability matrix \( P \), Perron vector is the normalized positive left eigenvector corresponding to the maximum eigenvalue \( \rho \), which is denoted as \( \phi \) [32]. Mathematically, it can be formulated as:

\[
\phi^P = \rho \phi \\
|\lambda_i| \leq \rho, \quad \text{for } \forall i \in [0, n-1] \\
\phi(v) > 0, \quad \text{for } \forall v \in V \\
\sum_v \phi(v) = 1,
\]

where \([\lambda_0, \lambda_1, \ldots, \lambda_{n-1}]\) are eigenvalues of \( P \). Notice that the Perron-Frobenius Theorem holds only for irreducible nonnegative matrixes, which calls for a directed graph to be strongly connected. For a general directed graph, the transition probability matrix is only nonnegative with weaker properties. A nonnegative matrix \( M \) has a nonnegative eigenvector and the corresponding nonnegative eigenvalue is no less than modulus of any eigenvalue. A detailed proof could be found in [39].

3) DIRECTED GRAPH LAPLACIAN [32]

Given a directed graph \( G \) with adjacent matrix \( A \), after defining transition probability matrix \( P \) and calculating the Perron vector \( \phi \), the definition of the Laplacian of a directed graph is given by:

\[
\mathcal{L} = I_n - \frac{\phi^{1/2} P \phi^{-1/2} + \phi^{-1/2} P^T \phi^{1/2}}{2}.
\]
\( \Phi \) is the diagonal matrix expanded by vector \( \phi \), i.e., \( \Phi(v, v) = \phi(v) \), and \( P^* \) denotes the conjugated transpose matrix of \( P \). There is another version of Laplacian called the combinatorial Laplacian \( \mathcal{L}_{com} \), defined as:

\[
\mathcal{L}_{com} = \Phi - \frac{\Phi P + P^* \Phi}{2}.
\]  

### III. METHODOLOGY

In this section, we propose a fast directed graph convolutional network model which is called FDGCN for semi-supervised node classification. At first, the inspiration of FDGCN is introduced. Then, detailed derivations for the directed graph convolution operator is given. Based on the derived convolution operator, a specific model is designed for semi-supervised node classification. Subsequently, it is found that FDGCN can unify GCN as a special case. Finally, a graph generating model MMSBM is integrated to further promote FDGCN.

#### A. INSPIRATION

In graph learning tasks, there are two kinds of information we can take advantage of. One kind of information is the feature vectors of nodes, known as the input matrix \( X \in \mathbb{R}^{n \times m} \), where each row is a node’s feature vector. Another kind of information is the structural information of graph, known as the adjacent matrix \( A \in \mathbb{R}^{n \times n} \). How to combine these two kinds of information reasonably is the key to succeed in node classification tasks.

Considering a pair of nodes in a graph \( G = (V, E) \), there are two cases in undirected scenario but four cases in directed scenario, which is shown as Fig. 1. Obviously, directed edges carry more information than undirected ones. In fact, there are so many scenarios in which a directed graph is more appropriate, such as social networks, citation networks, etc. Therefore, it would be helpful if we take the direction information into account in graph learning tasks.

\[
\begin{align*}
\text{undirected} & \quad \begin{array}{c}
\begin{array}{c}
\text{undirected}
\end{array}
\end{array} \quad \begin{array}{c}
\begin{array}{c}
\text{undirected}
\end{array}
\end{array} \\
\begin{array}{c}
\begin{array}{c}
\text{undirected}
\end{array}
\end{array} \quad \begin{array}{c}
\begin{array}{c}
\text{undirected}
\end{array}
\end{array}
\end{align*}
\]

\[
\begin{align*}
\text{directed} & \quad \begin{array}{c}
\begin{array}{c}
\text{directed}
\end{array}
\end{array} \quad \begin{array}{c}
\begin{array}{c}
\text{directed}
\end{array}
\end{array} \\
\begin{array}{c}
\begin{array}{c}
\text{directed}
\end{array}
\end{array} \quad \begin{array}{c}
\begin{array}{c}
\text{directed}
\end{array}
\end{array}
\end{align*}
\]

#### FIGURE 1. Edge information between a node pair.

### B. DIRECTED GRAPH CONVOLUTION OPERATOR

To construct a convolution operator on directed graphs, the directed graph Laplacian \( \mathcal{L} \) is mainly considered. Then its construction process, similar to that of GCN in Section II-B, will be elaborated in the following. Note that the transition probability matrix \( P \) is a nonnegative real matrix, which implies \( P^2 = P^T \). Therefore, \( \mathcal{L} \) is real symmetric and spectral theory can be exploited to construct convolution operator.

First we diagonalize directed graph Laplacian as \( \mathcal{L} = U \Lambda U^T \). Thus, the convolution can be rewritten and further approximated by \( K \)-th order Chebyshev polynomials:

\[
f_{\theta} * x = U f_{\theta}(\Lambda) U^T x \approx U \sum_{k=0}^{K} \theta_k T_k(\Lambda) U^T x = \sum_{k=0}^{K} \theta_k T_k(\tilde{\mathcal{L}}) x.
\]  

where \( \tilde{\mathcal{L}} = \frac{2}{\lambda_{\text{max}}} \mathcal{L} - I_n \) is the rescaled form, mapping eigenvalues from \([0, \lambda_{\text{max}}]\) to \([-1, 1]\). Take \( K = 1 \), then the above equation can be simplified to:

\[
f_{\theta} * x \approx \theta_0 x + \theta_1 \left( \frac{2}{\lambda_{\text{max}}} \mathcal{L} - I_n \right) x = \left( \theta_0 + \frac{2-\lambda_{\text{max}}}{\lambda_{\text{max}}} \theta_1 \right) x - \frac{2}{\lambda_{\text{max}}} \theta_1 \left( \Phi^{1/2} P \Phi^{-1/2} + \Phi^{-1/2} P^T \Phi^{1/2} \right) x.
\]  

In order to ignore the constant term of \( x \) and decrease number of free parameters, let \( \theta_0 = \frac{\lambda_{\text{max}} - 2}{\lambda_{\text{max}}} \theta_1 \) and \( \theta = \frac{2}{\lambda_{\text{max}}} \theta_1 \), we can get:

\[
f_{\theta} * x \approx \theta \frac{\Phi^{1/2} P \Phi^{-1/2} + \Phi^{-1/2} P^T \Phi^{1/2}}{2} x.
\]  

#### FIGURE 1. Edge information between a node pair.

Since \( \lambda_{\text{max}} \) is constant for a fixed Laplacian \( \mathcal{L} \) in a fixed graph, the equations about \( \theta_0 \) and \( \theta_1 \) can hold for the entire graph convolutional network. Thus only one parameter is needed for the entire filtering process.

The convolution in Eq. 10 is concise. However, the evaluation of Perron vector \( \phi \) has polynomial time complexity, which would make the convolution time-consuming, especially for large-scale graphs. To address the time consuming problem, we fix \( \phi \) as \( \left( \frac{1}{\sqrt{n}}, \ldots, \frac{1}{\sqrt{n}} \right)^T \), which is equal to the Perron vector for a directed regular graph. The reason why this special case could work for general directed graphs would be explained later in Section IV-A. Then, we can rewrite a decomposition-free convolution:

\[
f_{\theta} * x \approx \theta \frac{I_n - \sqrt{I_n} P (\sqrt{I_n} P + P^T \sqrt{I_n}) + (\sqrt{I_n} P + P^T \sqrt{I_n}) I_n}{2} x = \frac{\theta + \theta P^T}{2} x.
\]

The transition probability matrix \( P \) is derived from adjacent matrix \( A \). For a node in a general directed graph, of which the out degree may be zero, the corresponding row of \( P \) would sum up to zero. This does not conform to the definition of transition probability matrix. Therefore, we add a self-loop to each node to make sure that \( P \) is correctly constructed. Now the convolution process becomes:

\[
f_{\theta} * x \approx \theta \frac{P + P^T}{2} x = \theta \frac{\bar{\mathcal{D}}_{\text{out}}^{-1}(A + I_n) + (A + I_n)^T \bar{\mathcal{D}}_{\text{out}}^{-1}}{2} x \\
\rightarrow \theta \frac{\bar{\mathcal{D}}_{\text{out}}^{-1/2} \bar{\mathcal{D}}_{\text{out}}^{-1/2}}{2} x.
\]
where $\tilde{A} = A + I_n$ and $\tilde{D}_{out}(i, i) = \sum_j \tilde{A}_{ij}$. Further by replacing $D^{-1/2}AD^{-1/2}$ one, the final directed graph convolution operator can be obtained as a function of $A$, $D$, and $\tilde{D}_\text{in}$, i.e., $C_{DG}(A) = \frac{\tilde{D}_{out}(\tilde{A} + \tilde{A}^T)\tilde{D}_{out}}{2}$. The explanation of this replacing operation can be found in Section IV-B.1.

**C. FDGCN MODEL FOR SEMI-SUPERVISED NODE CLASSIFICATION**

In this section, we design a FDGCN model for semi-supervised node classification task. As we will explain in Section IV-A, the directed graph convolution operator $D_{DG}(A)$ aggregates features of neighbor nodes for each node with different weighted ratios. Specifically, the weighted ratio of aggregating node $v$ to node $u$ is related to the out-degree of both $u$ and $v$. In order to make full use of the direction information of edges and thoroughly integrate graph structure, we also take into consideration in-degree parameters [40]. As a consequence, a directed graph convolutional layer would consist of two directed graph convolution operators, $D_{DG}(A)$ and $D_{DG}(A^T)$. A FDGCN layer with input signal $X \in \mathbb{R}^{n \times c}$ of $c$ channels can be formulated as:

$$
Y = \frac{1}{2} \left( D_{out}^{-1/2} (\tilde{A} + \tilde{A}^T) D_{out}^{1/2} \right) X W_1 \\
+ \frac{1}{2} \left( D_{in}^{-1/2} (\tilde{A}^T + \tilde{A}) D_{in}^{1/2} \right) X W_2.
$$

(13)

Here $\tilde{D}_{in}(j, j) = \sum_i \tilde{A}_{ij}$ and $W_1, W_2 \in \mathbb{R}^{c \times h}$, where $h$ is the hidden node number of a layer. This two directed graph convolution operators would offer more capacity to the model. The directed graph convolution is complexity friendly, as there is only one multiplication of a sparse matrix with a dense matrix which can be efficiently implemented. The time complexity of a directed graph convolutional layer is $O(|E|ch)$, which is linear with the number of edges. If we regard undirected graphs as directed ones, it holds that $A = A^T$ and $D_{in} = D_{out}$. In this case, it can be found that Eq. 13 becomes $Y = D^{-1/2} \tilde{A} D^{-1/2} X(W_1 + W_2)$, which owns exactly the same form of GCN as Eq. 3.

The FDGCN model for semi-supervised node classification consists of two FDGCN layers, hidden layer and output layer. The number of layers is the optimal value got from a hyperparameters searching process. It is also consistent with the conclusion of [23]. As shown in Fig. 2, the hidden layer consists of $h$ units, each unit is a $n$-dimensional vector, calculated from the input features $X \in \mathbb{R}^{n \times c}$. The output layer has $l$ units, where $l$ is the number of categories. We use ReLU [41] as activation function, and the whole forward model can be formulated as Eq. 14 as shown at the bottom of the next page, where $D_{DG}^1 = D_{DG}(A)$ and $D_{DG}^2 = D_{DG}(A^T)$. The predicted label of node $i$ would be $Z_i = \arg \max_j Y_{ij}$, where $Y_{ij}$ is the $j$-th element of $l$-dimensional vector $Y_i$ for node $i$. Here $Z_i$ is expressed as one-hot vector. The corresponding loss function consists of cross-entropy over labeled examples and L2 regulation term of weights in first layer:

$$
F_{\text{loss}} = - \sum_{i \in \mathcal{S}_{\text{train}}} \sum_{j=1}^{l} Z_{ij} \log Y_{ij} + \eta \sum_{ij} |W_{ij}^{(0)}|^2.
$$

(15)

$S_{\text{train}}$ is the training set with labels and $\eta$ is the weight decay coefficient. $Z_{ij}$ represents for the $j$-th element of one-hot vector for the label of node $i$.

**D. FDGCN MODEL WITH MMSBM**

It should be emphasized that uncertainty of graph has been confirmed beneficial to promote GCN in [24]. In order to enhance FDGCN, mixed membership stochastic block model (MMSBM) [34] is considered to generate random graphs to give FDGCN extra training epochs. Firstly, FDGCN is trained on original graph $G_{\text{original}}$ with $N_{\text{pretrain}}$ epochs, where $X$ denotes the input signal and $Z_{\text{train}}$ denotes the labels in the training set. Then initialize MMSBM with $N_p$ iterations of inference. Secondly, FDGCN is trained on generated random directed graph for $N_{\text{extra}}$ epochs. Every random directed graph is generated by MMSBM with undirected form and turned into directed form subsequently. Finally, after all training process, FDGCN is performed on the test set doing inference and we get the predicted labels $Z_{\text{test}}$. The details are shown in Algorithm 1.
Algorithm 1 MMSBM-FDGCN

Input: \( G_{\text{original}}, X, Z_{\text{train}} \)

Output: \( Z_{\text{test}} \)

1. Train FDGCN on \( G_{\text{original}} \) with \( N_{\text{pretrain}} \) epochs;
2. Initialize MMSBM with \( N_p \) iterations of inference.
3. for \( i = 1 : N_{\text{extra}} \) do
4. Generate random undirected graph \( G_i \) with MMSBM.
5. Turn \( G_i \) into directed form \( G'_i \).
6. Train FDGCN on \( G'_i \) with one epoch.
7. if \( i \% N_{\text{interval}} = 0 \) then
8. Perform \( N_b \) iterations of inference to update MMSBM.
9. end
10. Do inference process of FDGCN to get \( Z_{\text{test}} \).

IV. ANALYSIS

In this section, we will take a deep insight into FDGCN and find out why it could work. As a consequence, we find that the convolution operator of FDGCN is designed as a spectral-based operator, but acts as a kind of spatial aggregation. From this spatial aggregation point of view, we can explain filtering mechanisms of many existing GCNN models and find relationship between them.

A. SPATIAL AGGREGATION POINT OF VIEW

As shown in Eq. 12, the directed graph convolution operator \( C_{DG}(A) \) consists of two parts, which are denoted by \( C_{DG1} = D_{out}^{-1/2}A\hat{D}_{out}^{-1/2} \) and \( C_{DG2} = \hat{D}_{out}^{-1/2}A\hat{D}_{out}^{-1/2} \), respectively.

Then we have \( C_{DG}(A) = \frac{C_{DG1} + C_{DG2}}{2} \). The value of each element of \( C_{DG1} \) and \( C_{DG2} \) can be respectively written as:

\[
C_{DG1}(u, v) = \begin{cases} 
\frac{1}{d_{out} + 1} & \text{if } u = v, \\
\frac{1}{\sqrt{(d_{out} + 1)(d_{out} + 1)}} & \text{if } A(u, v) \neq 0, \\
0 & \text{otherwise.}
\end{cases}
\]  

(16)

\[
C_{DG2}(u, v) = \begin{cases} 
\frac{1}{d_{out} + 1} & \text{if } u = v, \\
\frac{1}{\sqrt{(d_{out} + 1)(d_{out} + 1)}} & \text{if } A(v, u) \neq 0, \\
0 & \text{otherwise.}
\end{cases}
\]  

(17)

Next, we operate \( C_{DG} \) on input signals \( X \in \mathbb{R}^{n \times c} \), where \( X \) is denoted as the \( v \)-th row of \( X \). Consider the generated output of node \( u \) as an example:

\[
Y(u) = (C_{DG1} + C_{DG2})X(u) = \frac{1}{2} \left( \frac{1}{\sqrt{d_{out} + 1}} \sum_{u \rightarrow v, u} X_v + \frac{1}{\sqrt{d_{out} + 1}} \sum_{v \rightarrow u, v} X_v \right).
\]  

(18)

As we can see in Eq. 18, operating convolution on node \( u \) is equal to aggregating information of neighbor nodes including itself. The information of neighbor node \( v \) is aggregated with a weighted ratio, which is related to out-degree of both node \( u \) and node \( v \). From Eq. 16 and Eq. 17, it can be concluded that \( C_{DG1} \) and \( C_{DG2} \) are the weighted ratio matrixs of out-linked neighbor nodes and in-linked neighbor nodes, respectively.

Now looking back to our FDGCN layer, it is quite reasonable of taking into consideration both out-degree and in-degree. As the directed graph convolution operator \( D_{DG}(A) \) acts as a weighted ratios matrix in aggregation process, the actual effect of the Perron vector in Eq. 10 is to give all nodes an importance ranking. Since each node is already treated differently in calculating weighted ratios, treating each node equally by fixing \( \phi \) as \( \frac{1}{n} \) would not affect the capability of FDGCN.

B. RELATION WITH OTHER GCNN MODELS

With spatial point of view, we can further understand many previous GCNN models. Three models are selected, i.e., random walk [42], GCN [21] and graph attention network(GAT) [12]. Random walk and GAT are both spatial-based GCNN models while GCN is classic spectral-based GCNN model. Note that random walk adopts the transition probability matrix defined in directed graph Laplacian.

1) RANDOM WALK

Here a undirected graph \( G \) with self loop for each node is considered. The corresponding transition probability matrix is \( P = \hat{D}^{-1}(A + I_n) \), whose element can be expressed as:

\[
P(u, v) = \begin{cases} 
\frac{1}{d_u + 1} & \text{if } u = vorA(u, v) \neq 0, \\
0 & \text{otherwise.}
\end{cases}
\]  

(19)

Correspondingly, the output of node \( u \) would be:

\[
Y(u) = (PX)(u) = \frac{1}{d_u + 1} \sum_{v \sim u} X_v.
\]  

(20)

It can be observed that the random walk also aggregates information of node \( u \) and its neighbor nodes, but simply...
averages them to get the output. That exactly gives explanation to the replacing operation in Eq. 12, as this replacing operation would take into consideration more graph structure information by calculating weighted ratios with degrees of the node and its neighbor nodes.

2) GCN
The GCN convolution operator has the form of $C_G = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$, the element is:

$$C_G(u, v) = \begin{cases} 1 & \text{if } u = v, \\ \frac{1}{\sqrt{(d_u + 1)(d_v + 1)}} & \text{if } A(u, v) \neq 0, \\ 0 & \text{otherwise}. \end{cases} \quad (21)$$

Operating on input $X$, we can get output of node $u$ as:

$$Y(u) = (C_G X)(u) = \frac{1}{\sqrt{d_u + 1}} \sum_{v \sim u, u} X_v \quad (22)$$

We can see that the GCN convolution aggregates information of node $u$ and its neighbor nodes, with weighted ratios related to degrees of both linked nodes. Rethinking the renormalized trick of GCN as $I + \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$ → $\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$, after renormalization, the convolution process would treat node $u$ and its neighbor nodes equally. This is the reason why renormalization trick could work.

3) GAT
A GAT convolution operator working on node $u$ is inherently an aggregation of information of its neighbor nodes weighted by learned attention coefficients. By denoting $A$ as the attention coefficient matrix, we can express a one-head GAT convolution operator operating on input $X$ as:

$$Y(u) = (AX)(u) = \sum_{v \sim u, u} a_{uv} X_v \quad (23)$$

It is shown that GAT layer operation is very similar to GCN and FDGCN. The main difference is that GCN and FDGCN both derive the weighted ratio matrix with the help of domain knowledge, while GAT get the weighted ratio matrix by studying. This would give the model more capacity but also brings in more parameters, which makes it hard for training. With a sharing strategy, GAT can obtain the $n \times n$ attention coefficient matrix $A$ with $O(n)$ free parameters.

To sum up, all these models are spatial aggregation models. They all aggregate the features of neighbor nodes, and treat each neighbor node with different weights ratio according to its strategy. From this point of view, we get better understanding on these models, which gives good explanation to derivations of the proposed model. Focusing on two convolution operators of GCN and FDGCN, they are both first-order spatial aggregation operators. As shown in Fig. 3, convolution operators of GCN consider the degrees of a node and its neighbor nodes, while convolution operators of FDGCN consider the out-degrees to take advantage of direction information carried by edges. In our FDGCN model, a directed graph convolutional layer considers both in-degree and out-degree. Moreover, it treats the in-linked nodes and out-linked nodes differently, which gives the FDGCN more fitting capability.

V. EXPERIMENTS
In this section, the performance of the proposed FDGCN is evaluated for semi-supervised node classification task on three datasets. Experiments also provide comprehensive comparison against several state-of-the-art GCNN models, including GCN [21], GAT [12] and Bayesian GCN [24]. Note that Bayesian GCN is an improved method based on GCN, which utilizes graph uncertainty to significantly promote performance of classification. Inspired by Bayesian GCN, we exploit uncertainty of graphs and propose an integrated model, named as MMSBM-FDGCN, which gives FDGCN extra training epochs with graphs generated by MMSBM.

A. DATASETS AND SETUPS
The selected three datasets are well-known citation datasets: Cora, CiteSeer, and Pubmed [43]. The datasets used are processed from the original data, not the processed datasets [44] usually used in semi-supervised node classification experiments, which treat graphs undirected. Each dataset is divided into test set with 1000 instances and training set. For a training set, the number of labeled instances should be set same within each class. Specifically, we use three split setup, i.e., 5, 10 and 20 labeled instance per class, for training set in experiments. As we cannot recover the adjacent matrix of directed form with the fixed split [44] used in previous work, all results displayed in this section are evaluated from random split. Table 1 shows basic statistics of datasets. Undirected edges are less than directed ones due to two-way edges and self-loops. Note that Citeseer has 3312 nodes, which is different from the fixed split (3327) due to data missing. We also demonstrate degree percentile of all datasets for a better understanding of graph structure, which can be seen in Table 2.

The hyperparameters of FDGCN and MMSBM-FDGCN are same for all three datasets with: 2 layers, 64 hidden

1https://linqs.soe.ucsc.edu/data
TABLE 1. Datasets statistics.

|          | Cora | Citeeseer | Pubmed |
|----------|------|-----------|--------|
| Nodes    | 2708 | 3312      | 19717  |
| Edges(directed) | 5429 | 4715      | 44338  |
| Edges(undirected) | 5278 | 4598      | 44324  |
| Features | 1433 | 3703      | 500    |
| Classes  | 7    | 6         | 3      |
| Degree   | 3.89 | 2.77      | 4.49   |
| Out/In – degree | 2.00 | 1.42      | 2.24   |

TABLE 2. Datasets degree percentile.

| Percentile | 0  | 25 | 50 | 75 | 100 |
|------------|----|----|----|----|-----|
| Cora       |    |    |    |    |     |
| In – degree| 0  | 1  | 2  | 166|     |
| Out – degree| 0  | 1  | 2  | 3  | 5   |
| Degree     | 1  | 2  | 3  | 5  | 168 |
| Citeeseer  |    |    |    |    |     |
| In – degree| 0  | 0  | 0  | 0  | 99  |
| Out – degree| 0  | 0  | 0  | 1  | 2   |
| Degree     | 1  | 1  | 2  | 3  | 99  |
| Pubmed     |    |    |    |    |     |
| In – degree| 0  | 0  | 0  | 0  | 130 |
| Out – degree| 0  | 0  | 0  | 1  | 171 |
| Degree     | 1  | 1  | 2  | 4  | 171 |

TABLE 3. Cora results.

| Labels per class | 5    | 10   | 20   |
|------------------|------|------|------|
| GCN              | 70.0 ± 3.5   | 76.1 ± 2.3   | 79.8 ± 1.4   |
| GAT              | 70.4 ± 3.7   | 76.6 ± 2.8   | 79.9 ± 1.8   |
| BayesianGCN      | 74.4 ± 2.7   | 77.5 ± 1.7   | 80.1 ± 1.7   |
| FDGCN            | 69.7 ± 3.4   | 76.1 ± 1.9   | 80.0 ± 1.1   |
| MMSBM – FDGCN    | 74.4 ± 3.5   | 77.5 ± 2.1   | 80.4 ± 1.3   |

TABLE 4. Citeeseer results.

| Labels per class | 5    | 10   | 20   |
|------------------|------|------|------|
| GCN              | 69.5 ± 3.7   | 74.0 ± 3.3   | 77.4 ± 2.2   |
| GAT              | 68.0 ± 4.8   | 72.6 ± 3.6   | 76.4 ± 3.0   |
| BayesianGCN      | 70.3 ± 3.2   | 73.6 ± 2.6   | 76.1 ± 2.8   |
| FDGCN            | 68.9 ± 3.3   | 73.6 ± 3.7   | 77.4 ± 2.4   |
| MMSBM – FDGCN    | 70.0 ± 3.3   | 73.3 ± 2.9   | 75.1 ± 2.8   |

TABLE 5. Pubmed results.

| Labels per class | 5    | 10   | 20   |
|------------------|------|------|------|
| GCN              | 69.5 ± 3.7   | 74.0 ± 3.3   | 77.4 ± 2.2   |
| GAT              | 68.0 ± 4.8   | 72.6 ± 3.6   | 76.4 ± 3.0   |
| BayesianGCN      | 70.3 ± 3.2   | 73.6 ± 2.6   | 76.1 ± 2.8   |
| FDGCN            | 68.9 ± 3.3   | 73.6 ± 3.7   | 77.4 ± 2.4   |
| MMSBM – FDGCN    | 70.0 ± 3.3   | 73.3 ± 2.9   | 75.1 ± 2.8   |

In practice, we evaluate all these models implemented with TensorFlow [47] on a 32-core CPU server with 192GB RAM. The specific CPU is Inter(R) Xeon(R) CPU E5 – 2670 0 @ 2.60GHz.

B. RESULTS ANALYSIS

It should be noted that GCN and GAT both use validation set to select better model for classification. However in real semi-supervised scenario, there is no enough labeled data for model optimization. Therefore, no validation set (with labels) is used in whole evaluation.

For all the results in Table 3, 4 and 5, the highest accuracy of all these models in each column is highlighted in bold, while the best result of GCN, GAT and FDGCN in each column is underlined. It can be easily observed that FDGCN outperforms GCN and GAT in most cases. Besides, extra training epochs with MMSBM would usually lead to significant improvement and get the overall highest accuracy. To be specific, FDGCN performs better than GCN and GAT in all three split settings on Citeeseer and in the 20 labels per class setting on all three datasets, while MMSBM-FDGCN achieves overall highest accuracy in all three split settings on Citeeseer and in the 20 labels per class setting on Citeeseer and Cora.

There are bad cases for both FDGCN and MMSBM-FDGCN. As illustrated in [24], Pubmed dataset has a quite low intra-community density and a heavy-tailed degree distribution, which can also be seen in Table 2. In this situation, MMSBM becomes a bad choice for generating graphs, resulting in accuracy decrease for both Bayesian GCN (compared with GCN) and MMSBM-FDGCN (compared with FDGCN). The distributions of in-degree and out-degree of Pubmed dataset are more heavy-tailed, where most nodes own zero in-degree and out-degree. In this case, the effect of self-loop added on each node will dominate in weighted ratios of spatial aggregation. This is the reason why FDGCN has worse results on Pubmed than GCN. We notice that MMSBM-FDGCN gets a really bad performance in 5 labels per class setting on Citeeseer, with large performance degradation compared with FDGCN. As Citeeseer dataset has heavy-tailed distributions of both in-degree and out-degree, the FDGCN tends to overfit with very few labeled instances, the extra training epochs would make it worse.

To sum up, FDGCN has powerful feature extraction capability for directed graphs. It seems that FDGCN would become unstable on datasets with heavy-tailed distributions of in-degree and out-degree, which implies that...
C. Li et al.: Scalable GCNs With Fast Localized Spectral Filter for Directed Graphs

TABLE 6. Complexity analysis.

| Model          | Complexity (inference) | Parameters (operator) | Training Time (one epoch) | Training Time (total) |
|----------------|------------------------|-----------------------|--------------------------|-----------------------|
| GCN            | $O(|\mathcal{E}| \cdot m)$ | $O(1)$               | 0.07(s)                  | 13.57(s)              |
| GAT            | $O(|\mathcal{E}| \cdot m)$ | $O(n)$               | 4.96(s)                  | 996.98(s)             |
| BayesianGCN    | $O(|\mathcal{E}| \cdot m)$ | ~                     | ~                        | 1431.18(s)            |
| FDGCN          | $O(|\mathcal{E}| \cdot m)$ | $O(1)$               | 0.08(s)                  | 22.41(s)              |
| MMSBM – FDGCN  | $O(|\mathcal{E}| \cdot m)$ | ~                     | ~                        | 1485.01(s)            |

TABLE 7. Training time of FDGCN.

| Model          | Nodes | Edges | Features | Wall-clock Training Time |
|----------------|-------|-------|----------|--------------------------|
| FDGCN          | 22.41(s) | 25.54(s) | 63.48(s) | 251.98(s)                |

TABLE 8. Comparison among recent GCNN models on Citeseer dataset with 20 labels per class.

| Model          | Micro-F1(Acc) | Macro-F1   |
|----------------|---------------|------------|
| GCN            | 68.8 ± 1.8    | 65.0 ± 1.6 |
| HGCN           | 67.4 ± 1.8    | 63.7 ± 2.0 |
| TGCN           | 69.2 ± 1.8    | 65.3 ± 1.8 |
| FDGCN          | 69.5 ± 1.8    | 65.9 ± 1.7 |

D. ADDITIONAL EXPERIMENT

To better evaluate the performance of FDGCN model, some additional experiments are conducted. Several recent GCNN models are selected, including GCN [21], HGCN [27], TGCN [22] and FDGCN proposed in this paper. All experiments are conducted on Citeseer dataset with 20 labels per class and other setups are same as Section V-A. All hyperparameters of models have the same set as their original papers. Mean Micro-F1 (i.e. accuracy) and Macro-F1 score of each model is demonstrated in Table 8, which is different from the previous experiments. As shown in Table 8, FDGCN model is superior than other models in both Micro-F1 and Macro-F1 score. Additional comparison experiments with recent models and new measurement demonstrate the superiority of FDGCN model.

VI. CONCLUSIONS

In this paper we propose a scalable graph convolutional network with fast localized spectral filter named FDGCN, which is targeted for directed graphs. It turns out that FDGCN can generalize GCN as a particular instance thereof. A fast localized convolution operator in FDGCN is constructed by integrating 1st-order Chebyshev polynomials approximation of spectral filter, together with the approximation by fixing Perron vector. With these approximations, FDGCN can also achieve scalability since the computational complexity of convolution operation is linear with the size of edge set. Meanwhile, FDGCN is easy to be trained as the number of parameters of each convolution operator is $O(1)$. Experiments for semi-supervised node classification tasks are conducted to evaluate the performance of FDGCN model. In most cases our model can outperform other models and we also give explanation why it dose not work so well in some cases. In future, we will try to apply FDGCN model for different graph learning tasks such as link prediction, to further explore its capability.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., vol. 2, 2012, pp. 1097–1105.
[2] 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.
[3] W. Cai and Z. Wei, “PriGAN: Generative adversarial networks for pluralistic image inpainting,” IEEE Access, vol. 8, no. 2, pp. 48451–48463, 2020.
XIAOWEI QIN received the B.S. and Ph.D. degrees from the Department of Electrical Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 2000 and 2008, respectively. Since 2014, he has been a Member of Staff with the Key Laboratory of Wireless-Optical Communications of Chinese Academy of Sciences, USTC. His research interests include optimization theory, service modeling in future heterogeneous networks, and big data in mobile communication networks.

XIAODONG XU received the B.S. and Ph.D. degrees from the Department of Electrical Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 2000 and 2007, respectively. He is currently a Member of Staff with the Key Laboratory of Wireless-Optical Communications of Chinese Academy of Sciences, USTC. His current research interests include communication signal processing, blind signal processing, and machine learning in wireless communications.

DUJIA YANG received the B.S. degree in microelectronics from the Hefei University of Technology, Hefei, China, in 2013. He is currently pursuing the Ph.D. degree with the Wireless Information Network Laboratory, University of Science and Technology of China, Hefei. His current research interests include wireless communications, big data analysis, and deep learning.

GUO WEI received the B.S. degree in electronic engineering from the University of Science and Technology of China (USTC), Hefei, China, in 1983, and the M.S. and Ph.D. degrees in electronic engineering from the Chinese Academy of Sciences, Beijing, China, in 1986 and 1991, respectively. He is currently a Professor with the School of Information Science and Technology, USTC. His current research interests include wireless and mobile communications, wireless multimedia communications, ultrawideband communication systems, and wireless information networks.