LFGCN: Levitating over Graphs with Levy Flights

Yuzhou Chen, Yulia Gel, Konstantin Avrachenkov

To Appear in the 2020 IEEE International Conference on Data Mining (ICDM).

LFGCN: Levitating over Graphs with Levy Flights

Yuzhou Chen
Department of Statistical Science
Southern Methodist University
Dallas, USA
yuzhouc@smu.edu

Yulia R. Gel
Department of Mathematical Sciences
University of Texas at Dallas
Richardson, USA
ygl@utdallas.edu

Konstantin Avrachenkov
Network Engineering and Operations
Inria Sophia Antipolis
Sophia Antipolis, France
k.avrachenkov@inria.fr

***Abstract*—Due to high utility in many applications, from social networks to blockchain to power grids, deep learning on non-Euclidean objects such as graphs and manifolds, coined Geometric Deep Learning (GDL), continues to gain an ever increasing interest. We propose a new Lévy Flights Graph Convolutional Networks (LFGCN) method for semi-supervised learning, which casts the Lévy Flights into random walks on graphs and, as a result, allows both to accurately account for the intrinsic graph topology and to substantially improve classification performance, especially for heterogeneous graphs. Furthermore, we propose a new preferential P-DropEdge method based on the Girvan-Newman argument. That is, in contrast to uniform removing of edges as in DropEdge, following the Girvan-Newman algorithm, we detect network periphery structures using information on edge betweenness and then remove edges according to their betweenness centrality. Our experimental results on semi-supervised node classification tasks demonstrate that the LFGCN coupled with P-DropEdge accelerates the training task, increases stability and further improves predictive accuracy of learned graph topology structure. Finally, in our case studies we bring the machinery of LFGCN and other deep networks tools to analysis of power grid networks – the area where the utility of GDL remains untapped.**

***Index Terms*—graph-based semi-supervised learning, convolutional networks, Lévy flights, local graph topology**

I. Introduction

Adaptation of deep learning (DL) to graphs and other non-Euclidean objects has recently witnessed an ever increasing interest, leading to the new subfield of *geometric deep learning (GDL)*. In particular, geometric deep learning is an emerging direction in machine learning which generalizes concepts of deep learning for data in non-Euclidean spaces, e.g., graphs and manifolds, by bridging the gap between graph theory and deep neural networks [1, 2, 3].

Many such DL approaches for non-Euclidian objects are based on the idea of a convolution operation in the spectral domain with a suitably chosen nonlinear trainable filter [see an overview by 4]. As a result, node features are mapped into some Euclidian space. Next, graph filters are approximated with various finite order polynomials, e.g., Chebyshev polynomials [i.e., the ChebNet model family of 5, 6], Cayley transform [i.e., CayleyNet of 7] or the generalization of polynomial filters in a form of Auto-Regressive Moving Average (ARMA) models [8]. However, deep learning approaches based on approximation with finite order polynomials tend to be non-robust to even minor changes in the graph structure and to largely disregard the local graph topology which often plays the critical role for learning on heterogeneous graphs. In contrast, as noted by [8], one of the primary benefits of the ARMA filters over polynomial ones is that ARMA filters are not computed in the Fourier space induced by a graph Laplacian, and as a result, ARMA filters are local in the node space and enable to more flexibly and accurately capture the underlying graph topology.

Further advancing the localized approaches in GDL, we propose a new fractional generalized graph-based convolutional filter for semi-supervised learning which casts the Lévy Flights into random walks on graphs. As a result, our new Lévy Flights Graph Convolutional Network (LFGCN) method allows to more accurately account for the intrinsic local graph topology and to substantially improve classification performance, especially for heterogeneous graphs.

To fly or not to fly, and if to fly, why take a Lévy Flight? Lévy Flight is a random process with a scale-free, Lévy stable jump length distribution. Due to the scale-free character, throughout the graph exploration we move randomly according to a power-law distribution for hops, rather than with integer hops as in a standard random walk. As a result, Lévy Flight delivers more accurate and efficient search strategies, especially, in sparse environments, comparing to other types of random walks [9]. While superiority of Lévy Flights as a primary search strategy has been proven in a broad range of settings, utility of Lévy Flights in graph-based learning remains unexplored. Hence, Lévy Flights offer a new learning perspective with multiple advantages comparing to the currently available architectures. First, due to a fractal character, Lévy Flights combines both local graph exploration and long-range excursions, which reduces oversampling comparing to a normal random walk (i.e., lower probability to revisit the nodes we have already seen). Second, Lévy Flights allow to directly reach long-distance nodes without intervention of intermediate nodes. Third, based on average global time, Lévy Flights average return probability is of power-law distr., i.e. $p_0^{\gamma}(t) \sim t^{-1/2\gamma}$ and is lower than average return probability of a normal random walk $p_0^1(t) \sim t^{-1/2}$, thereby leading to more efficient graph exploration. Forth, Lévy Flights are known to exhibit a particularly high utility for unbalanced and directed data which can explain higher LFGCN accuracy we have obtained in directed networks.

In addition, to abate over-fitting and over-smoothing in GDL, we develop a new preferential P-DropEdge method based on censoring edge order statistics at each training epoch. Our P-DropEdge idea is inspired by the recent DropEdge algorithm of [10] and is rooted in nonparametric methods, specifically, various censoring schemes, for statistical inference

on order statistics [11]. In contrast to uniformly removing edges as in the recent DropEdge algorithm of [10], we follow the Girvan-Newman argument and target edges that tend to contribute more to the intrinsic graph topology. That is, we randomly remove edges with higher betweenness centrality, or the corresponding higher edge order statistics. The intuition is the following. In both P-DropEdge and DropEdge the goal is to introduce randomness in the network structure. If we are to learn international political networks with GDL, DropEdge largely tends to remove connections among individual citizens while P-DropEdge randomly censors collaboration links among Presidents and Prime Ministers. Removal of such targeted connection is likely to lead to higher perturbation effects. We investigate utility of the new P-DropEdge approach vs. DropEdge in conjunction with LFGCN and GMMN (the best performing baseline) of [12].

Significance of our contributions can be summarized as:

- we propose a new fractional generalized graph-based convolutional filter for semi-supervised learning which casts the Lévy Flights into random walks on graphs and, as a result, provides a more efficient exploration of graph structures. We develop a new Lévy Flights Graph Convolutional Network (LFGCN) method that substantially improves accuracy of node classification for graphs, outperforming 12 State-of-the-Art (SOTA) methods on 2 out 3 considered undirected networks and 10 SOTAs on all 4 considered directed networks.
- the proposed architecture of LFGCN uses three state-of-the-art operations – *gated max-average pooling*, residual block, and P-DropEdge. We provide an ablation study and investigate contribution of each component to the resulting classification accuracy as well as explore sensitivity of the overall system architecture to (hyper)parameter settings.
- we provide theoretical foundations behind the proposed LFGCN architecture and show that the proposed LFGCN architecture leads to significant gains in the training convergence and model output stability.
- the developed preferential P-DropEdge based on censoring of higher edge betweenness order statistics is shown to exhibit utility in other GCN methods and, hence, might be applicable in broader GDL settings.
- Last but not the least, while validating our LFGCN methodology, we bring the GDL concepts to the analysis of power grid networks, i.e., the area of critical societal importance where to the best of our knowledge, the GDL machinery has never been yet applied.

II. Related Work

Many earlier semi-supervised learning approaches on graphs, e.g., Gaussian mixture models, co-training, harmonic function, and label propagation, tend to employ only the label information (i.e., labeled instances) for training models based on the smoothness assumption over the labels [13] and to largely disregard the underlying graph structure. To enhance performance, several learning methods on graphs propose to incorporate intrinsic "graph-based" information by designing a classifying function via generalizing the normalized cut and adding a smooth function with respect to the intrinsic structure [14, 15]. An optimization framework of [16] generalizes these approaches by considering the above two methods as particular cases. However, the major criticism to these graph-based semi-supervised learning methods is that important information contained in graph edges is largely disregarded.

To address these limitations, [5] propose a formulation of convolutional neural networks (CNN) based on spectral graph theory – ChebNet. ChebNet employs approximation via finite order polynomials and is based on the Chebyshev expansion for fast filtering instead of the expensive eigen-decomposition. Graph Convolutional Networks (GCN) of [6] simplifies ChebNet while further addressing the gradient vanishing problem and reducing the number of optimization. Other related approaches to graph learning with deep neural networks include, for instance, mixture model networks (MoNet) [2], graph attention networks (GAT) [17], graph convolutional recurrent networks [18], dual graph convolutional networks [19], FastGCN [20], and simplified version of GCN [21]. By directly powering the graph Laplacian, GCN based on random walks such as approximate personalized propagation of neural predictions (APPNP) [22], variable power network (VPN) [23], and MixHop [24] can learn the relationships between multiple-hops neighborhood.

To extend the success of GCN on undirected graphs to directed graphs, MotifNet of [25] replaces the normalized Laplacian with the *motif Laplacian* in a multivariate polynomial filter, where the motifs information can help capture the network structure. Finally, the most recent approach of [8] provides more flexible responses than GCN by using parallel and periodic concatenations of the convolutional kernel via the ARMA filter. As a result, the ARMA approach which is applicable to both directed and undirected networks allows to more accurately incorporate the underlying local graph structure into the graph learning process. For a recent comprehensive overview of GCNs see [4].

III. Methodology

Consider a graph structure $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, W\}$, where $\mathcal{V}$ is a node set with cardinality $|\mathcal{V}|$ of N, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is an edge set. An $N \times N$-matrix W with entries $\{\omega_{ij}\}_{1 \le i,j \le N}$ represents the adjacency matrix of $\mathcal{G}$, that is, $\omega_{ij} \neq 0$ for any $e_{ij} \in \mathcal{E}$ and $\omega_{ij} = 0$, otherwise. For an undirected graph $\mathcal{G}$, $W = W^\top$. In reality, however, undirected graphs are often simplified representations of complex directed networks. If $\mathcal{G}$ is directed, we substitute W with $W' = (W^\top + W)/2$.

Let $Q, Q \in \mathbb{Z}_{>0}$ be the number of different node features associated each node $v \in \mathcal{V}$. Then, a $N \times Q$ feature matrix X serves as an input to an semi-supervised learning algorithm. To classify N data points into K classes (communities), we define a $N \times K$ label matrix Y such that $Y_{ik} = 1$ if vertex i is labeled as class k, and 0 otherwise. Here we refer to each column $Y_{\cdot k}$ of matrix Y as a *labeling function*. Finally, we define an $N \times K$ matrix F whose columns $F_{\cdot k}$ are referred to as *classification functions*.

A. Graph signal processing

Given the adjacency matrix W of $\mathcal{G}$, let D be the degree matrix where $d_{ii} = \sum_{j=1}^{N} w_{ij}$ and $L = U^\top \Lambda U$ be the Standard Laplacian matrix. Here $\Lambda = diag(\lambda_0, ..., \lambda_{N-1})$ and $U = [u_0, ..., u_{N-1}]$ is the matrix of eigenvectors.

In the following, we will revisit three popular semi-supervised learning methods - graph-based semi-supervised learning, fractional graph-based semi-supervised learning, and graph convolutional networks and gain new insights for improving their modeling capabilities.

Graph-based semi-supervised learning Graph-based semi-supervised learning (G-SSL) has received much attention as an alternative approach to the population paradigm of supervised learning in recent years. G-SSL develops a generalized optimization framework, which has three particular cases (i) the Standard Laplacian (SL); (ii) Normalized Laplacian (NL); (iii) PageRank (PR). The general idea of graph-based semi-supervised learning (G-SSL) is based on two widely used optimization frameworks. The first formulation, the SL based formulation [15] as follows:

$$\min_F \left\{ \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\|F_{i.}-F_{j.}\|^2 + \mu \sum_{i=1}^{N} d_i \|F_{i.}-Y_{i.}\|^2 \right\},$$

where d_{ii} is (i,i)-element in degree matrix D and w_{ij} represents the edge weight for edge e_{ij} in adjacency matrix W. For the second formulation, the NL based formulation [14], is as follows:

$$\min_F \left\{ \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} \left\| \frac{F_{i.}}{\sqrt{d_{ii}}} - \frac{F_{j.}}{\sqrt{d_{jj}}} \right\|^2 + \mu \sum_{i=1}^{N} \|F_{i.}-Y_{i.}\|^2 \right\}$$

The following lemma [16] asserts that the generalized optimization framework, i.e., G-SSL, which has as particular cases the two above mentioned formulations:

Lemma 1. *Let σ denote an alternative parameter on the power of degree matrix D whose entries are the degrees d_{ii}; and let $0 \le \sigma \le 1$. Then*

$$\min_F \left\{ \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij} \left| d_{ii}^{\sigma-1}F_{i.} - d_{jj}^{\sigma-1}F_{j.} \right|^2 + \mu \sum_{i=1}^{N} d_{ii}^{2\sigma-1} \|F_{i.}-Y_{i.}\|^2 \right\}.$$

The classification functions for the generalized semi-supervised learning are given by $F_{.k} = (1-\alpha)\big(I-\alpha D^{-\sigma}WD^{\sigma-1}\big)^{-1}Y_{.k}$.

Proof of **Lemma 1** is in Appendix VII. The optimization formulation $S(F)$ with the following expression:

$$\begin{aligned} S(F) &= \min_F \Big\{ 2F_{.k}^T D^{\sigma-1} L D^{\sigma-1} F_{.k} \\ &+ \mu (F_{.k}-Y_{.k})^T D^{2\sigma-1}(F_{.k}-Y_{.k}) \Big\}, \end{aligned} \tag{1}$$

where μ is a regularization parameter. Minimization of the 1st term in (1) corresponds to the idea that if two nodes are close in graph with respect to some metric, they should belong to the same class; and by minimizing the 2nd term we aim to bring the classification function $F_{.k}$ as close as possible to the labeling function $Y_{.k}$. Eq. (1) allows us to obtain the Standard Laplacian based formulation ($\sigma=1$), the Normalized Laplacian formulation ($\sigma = 0.5$), and PageRank formulation ($\sigma = 0$). Objective of the generalized optimization framework for G-SSL is a convex function and the corresponding classification function:

$$F_{.k} = \frac{1-\alpha}{I-\alpha D^{-\sigma}WD^{\sigma-1}}Y_{.k}, \quad \alpha = \frac{2}{2+\mu}, 1 \le k \le K. \tag{2}$$

By tuning the parameter σ on the power of degree matrix D, we can obtain three mentioned above particular semi-supervised learning methods:

$$\begin{aligned} \sigma &= \textbf{1 - SL}: F_{.k} = (1-\alpha)\big(I-\alpha D^{-1}W\big)^{-1}Y_{.k}, \\ \sigma &= \frac{\mathbf{1}}{\mathbf{2}} \textbf{ - NL}: F_{.k} = (1-\alpha)\Big(I-\alpha D^{\frac{-1}{2}}WD^{\frac{-1}{2}}\Big)^{-1}Y_{.k}, \\ \sigma &= \textbf{0 - PR}: F_{.k} = (1-\alpha)\big(I-\alpha WD^{-1}\big)^{-1}Y_{.k}. \end{aligned}$$

From above formulations, classification function F is a closed form solution based on the theory of random walks on graphs, which in turn provides connection to the probabilistic interpretation of G-SSL. Parameter α controls the strength of the ground truth label matrix Y in the generalized optimization framework.

Fractional graph-based semi-supervised learning To improve classification performance (in particular, fuzzy graphs and unbalanced labeled data) of G-SSL, fractional graph-based semi-supervised learning [26] embeds Lévy Flights into random walks on graphs by constructing from powers of the Laplacian matrix, i.e., the L^γ operator. This operation can be used to generate different transition probabilities (i.e., corresponding to stochastic adjacency matrix) based on different γ values. Intuitively, embedding Lévy Flights into random walks allows for better capturing mixing properties (i.e., dependence) in the data. Based on a fractional Laplacian matrix, $0 < \gamma \le 1$, the anomalous (fractional) diffusion processes on networks can be constructed from the spectra data and eigenvectors of the Laplacian matrix. The fractional powers of L allows Lévy random walks with long-range navigation on a network. For example, the long-range transitions on a network can directly move node u and node v with the transition probability $m_{u\to v}^{(\gamma)}$ through a random walker, where $m_{u\to v}^{(\gamma)}$ is an element in the fractional transition matrix $\mathbf{M}^{(\gamma)}$. Transition probability $m_{u\to v}^{(\gamma)}$ between any two nodes whose geodesic distance is not infinite can be summarized as follow:

$$m_{u\to v}^{(\gamma)} = \delta_{uv} - (L^\gamma)_{uv}/k_u^{(\gamma)}, \tag{3}$$

where δ_{uv} is the Kronecker delta, $k_u^{(\gamma)}$ denotes the fractional degree of the node u and $k_u^{(\gamma)} \equiv (L^\gamma)_{uu}$. Eq. (3) provides transition probabilities for the Lévy Flights. Unlike the standard random walk, the Lévy Flights can jump immediately over several hops in a graph. This feature enables Lévy Flights to be a very effective graph exploratory process. Lemma 2 makes this statement formal.

Lemma 2. *The Lévy flight defined by the normalized Laplacian has a shorter relaxation time (measure of the transience) in comparison with the original random walk.*

Proof of **Lemma 2** is in Appendix VIII. There is a price to pay for this: the typically sparse transition probability matrix becomes non-sparse. We can mitigate non-sparsity by taking a

reasonable number of principal singular eigenvectors or limiting the number of terms in the Taylor expansion. Through replacing the L operator with $L^\gamma = U^\top \Lambda^\gamma U$, the new optimization formulation $S^*(F)$ leaves us with the following expression:

$$\begin{aligned} S^*(F) &= \min_F \big\{ 2F_{\cdot k}^T D_\gamma^{\sigma-1} L^\gamma D_\gamma^{\sigma-1} F_{\cdot k} \\ &+ \mu (F_{\cdot k} - Y_{\cdot k})^T D_\gamma^{2\sigma-1} (F_{\cdot k} - Y_{\cdot k}) \big\}, \end{aligned} \tag{4}$$

where $(D_\gamma)_{ii} = (L^\gamma)_{ii}$.

Let $0 < \gamma \leq 1$, then the closed form solution for (4) can be obtain as follows: $F_{\cdot k} = (1-\alpha)\big(I - \alpha D_\gamma^{-\sigma} W_\gamma D_\gamma^{\sigma-1}\big)^{-1} Y_{\cdot k}$, for $k = 1,...,K$. Therefore, we can conclude three particular fractional semi-supervised learning methods like G-SSL:

$$\sigma = \mathbf{1} \text{ - } \mathbf{FSL}: F_{\cdot k} = (1-\alpha)\big(I - \alpha D_\gamma^{-1} W_\gamma\big)^{-1} Y_{\cdot k},$$
$$\sigma = \frac{\mathbf{1}}{\mathbf{2}} \text{ - } \mathbf{FNL}: F_{\cdot k} = (1-\alpha)\Big(I - \alpha D_\gamma^{\frac{-1}{2}} W_\gamma D_\gamma^{\frac{-1}{2}}\Big)^{-1} Y_{\cdot k},$$
$$\sigma = \mathbf{0} \text{ - } \mathbf{FPR}: F_{\cdot k} = (1-\alpha)\big(I - \alpha W_\gamma D_\gamma^{-1}\big)^{-1} Y_{\cdot k}.$$

B. Proposed Lévy Flights Graph Convolutional Network for semi-supervised node classification

Although both G-SSL and fractional G-SSL achieve comparable and consistent (low variance) performance on some datasets, e.g., Les Miserables, Wikipedia-math, and MNIST, these approaches consider only the given adjacency matrix W and the label matrix Y, without using the feature matrix X. This limitation is crucial, especially when dealing with datasets that not only exhibit a sophisticated topological graph structure but also provide node feature information, such as citation, biological, financial, and power grid networks. To address this limitation, there have been recently proposed many graph-based neural networks methods, e.g., graph convolutional networks (GCN), which use the feature matrix X instead of the label matrix Y and encode the graph structure by using neural network framework. Such graph-based neural networks have been shown to achieve impressive gains in semi-supervised learning performance on graphs. Next, we turn to discussing on how the idea of Lévy Flights can be incorporated to GCN, leading to the new Lévy Flights Graph Convolutional Network (LFGCN) for semi-supervised node classification.

Lévy Flights Graph Convolutional Network (LFGCN) The key idea behind our proposed method is Fractional Generalized Sigma-based (FGS) filter

$$g_{FGS}(\alpha,\sigma,\gamma) = \frac{1-\alpha}{I - \alpha D_\gamma^{-\sigma} W_\gamma D_\gamma^{\sigma-1}} = \frac{1-\alpha}{I - \alpha \tilde{L}}.$$

To avoid the inverse computations, we insert the Taylor series expansion into the FGS filter, resulting in:

$$g_{FGS}(\alpha,\sigma,\gamma) = (1-\alpha)\sum_{i=0}^{\infty} \big(\alpha \tilde{L}\big)^i, \quad 0 < \alpha,\sigma,\gamma \leq 1. \tag{5}$$

Empirically, it shows that $i = \lceil 4\alpha \rceil$ is enough to get a good approximation. We then obtain the general classification function by multiplying (5) by the feature matrix X:

$$\begin{aligned} \bar{\mathcal{X}} &= g_{FGS}(\alpha,\sigma,\gamma) X \\ &= (1-\alpha)\Big(\underbrace{X}_{\text{the 1st item}} + \underbrace{\alpha \tilde{L} X}_{\text{the 2nd item}} + \underbrace{\alpha^2 \tilde{L}^2 X}_{\text{the 3rd item}} + \cdots \Big), \\ &= (1-\alpha)(X')_i \end{aligned} \tag{6}$$

where $(X')_i = X + \alpha \tilde{L}(X')_{i-1}$, $(X')_0 = X$, $i \in \mathbb{Z}_{i \geq 0}$.

Convolutional layer During LFGCN training, the convolutional model needs to train parameters $(\mathbf{W}, \mathbf{b})$ of the graph filter, where the trainable graph filter scan the given input feature matrix into a series of feature maps with neurons. Thereby, we provide an implementation of (6) as a FGS convolutional layer:

$$H^{(t+1)} = \sigma\left((1-\alpha) \sum_{i=0}^{\infty} \Big(\alpha \tilde{L}\Big)^i H^{(t)} \mathbf{W}^{(t)} \right), \tag{7}$$

where $H^{(t+1)}$ is the hidden layer output matrix of activations in the t-th layer and $H^{(0)} = X$, $\sigma(\cdot)$ is the adopted activation function, and $\mathbf{W}^t$ is the trainable weight in the t-th layer. Furthermore, we bring the concept of the parallel system (PS) from the reliability theory to improve the consistency of our proposed method. A parallel system is a configuration such that the entire system functions as long as not all involved components in the system fail. Hence, the parallel system structure is more robust against noisy inputs, compared to a single system structure.

Lemma 3. *Let $\bar{X}_{FGS}$ be the output matrix P_1 from a pooling layer. Let $\mathcal{U} = \{1,2,...,N\}$ be a finite population such that each unit $i, i \in \mathcal{U}$ is associated with an output matrix $X_{FGS}^{(i)}$, $i = 1,...,N$. Then $\mathrm{Var}(\bar{X}_{FGS}) = (1 - n/N) S^2/n$, $n < N, n \in \mathbb{Z}_{>0}$, where $S^2 = \sum_{i=1}^{N} \big(X_{FGS}^{(i)} - \bar{X}_{FGS}^{U}\big)^2/(N-1)$.*

Suppose there are n components in a parallel system, with the probability of non-failure $P_R^{(i)}$ (where $i = 1, \cdots, n$) in a parallel system, then the reliability of this parallel system P_R^{PS} can be obtained with the following expression:

$$P_R^{PS} = 1 - \prod_{i=1}^{n} (1 - P_R^{(i)}).$$

Proof of **Lemma 3** is in Appendix IX. According to Lemma 3, the introduced concept of a parallel system allows for enhancing stability and reducing estimation variance up to order of n (i.e., $\mathrm{Var}(\bar{X}_{FGS}) = O(S^2/n)$). In this way, we establish both theoretical and practical guarantees for our proposed model to reach stable over a large set of hyperparameters, small datasets, and noisy labels based on this parallel implementation.

Pooling layer When implementing the form of pooling operation to aggregate information from the outputs of parallel FGS convolutional layer, instead of using some popular pooling functions such as max and average pooling, we apply the state-of-the-art pooling operation - *gated max-average pooling* [27] to capture the local and global information from all the nodes and graph structure. The rationale behind the *gated max-average pooling*, is that it considers "responsive" strategy (i.e., improving translation invariance and scale

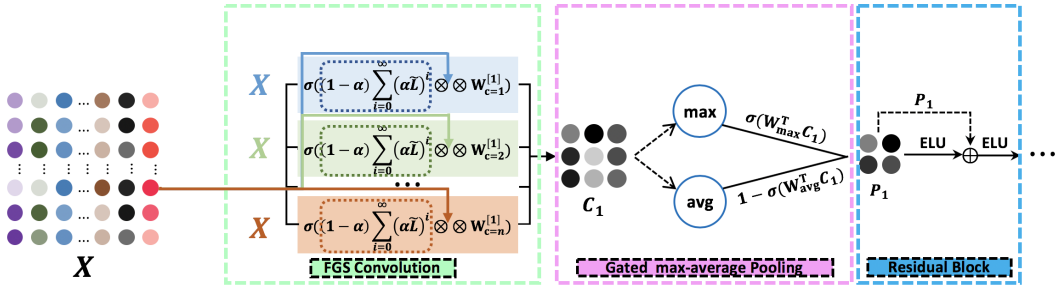

Fig. 1: Illustration Lévy Flights Graph Convolutional Network model. The input is the feature matrix X and the graph within dotted circle represents embedding Lévy Flights into random walks on graph (where L^γ is the Laplacian matrix L to a power γ). LFGCN architecture consists of three main components: (i) FGS convolutional layer with parallel structure; (ii) *gated max-average pooling* layer; (iii) activation block for residual learning.

invariance via considering input in each gating mask) based on the *mixed max-average pooling* equation. That is,

$$\begin{aligned} f_{\text{gated}}(X_{FGS}) &= \sigma\big(\mathbf{W}^\top X_{FGS}\big) f_{\max}(X_{FGS}) \\ &+ \big(1-\sigma\big(\mathbf{W}^\top X_{FGS}\big)\big) f_{\text{avg}}(X_{FGS}), \end{aligned}$$

where $\mathbf{W}$ is the trainable weight matrix, X_{FGS} is the output matrix from the parallel FGS convolutional layer after concatenation operation.

Residual building block Inspired by the seminal works of [28, 29] that implemented residual learning in a graph convolutional network, we apply a residual block (RB) by adding the skip connection after the pooling layer. One of the advantages of the residual learning is the *identity* mapping which provides a direct path for propagating information. When using the residual building block, we adopt a similar scheme as [30] to deal with the output of the pooling layer. Let $\mathcal{H}(x)$ be an underlying mapping and we cast it as $\mathcal{H}(x)=\mathcal{F}(x)+x$, where $\mathcal{F}(x)$ is the residual mapping, defined by $\mathcal{H}(x)-x$. That is, optimizing the residual mapping $\mathcal{F}(x)$ is easier than optimizing the direct mapping $\mathcal{H}(x)$ and helps to avoid the gradient vanishing problem during training. We use an exponential linear unit (ELU) in direct mapping and place a rectified linear unit (ReLU) after addition in our model.

P-DropEdge Motivated by the recent idea of message passing inference [i.e., DropEdge of 10], we develop a new preferential DropEdge approach called the *P-DropEdge* which is based on censoring higher edge betweenness order statistics. In particular, most recently [10] propose a flexible approach, the DropEdge algorithm which by uniformly randomly removing a certain proportion of edges from the input graph at each training epoch, allows to better prevent against over-fitting and to reduce the effect of over-smoothing. The rationale behind DropEdge on introducing more randomness and deformation into the data is intrinsically linked and complementary to the Dropout ideas of [31]. Our approach further advances DropEdge by targeting and randomly removing edges proportionally to their betweenness centrality, i.e., preferential edge dropout of higher edge betweenness order statistics. That is, first, our idea is based on the Girvan-Newman argument of focusing on edges which tend to play a higher role in the underlying network topology [32]. Second, dropout of higher edge betweenness order statistics may be viewed as a variant of recently proposed non-uniform censoring schemes for generalized order statistics in reliability theory which are shown to deliver more robust parameter estimates in heterogeneous probability distributions [33].

Definition 1.. *(Edge Order Statistics) Given a input graph $\mathcal{G}=\{\mathcal{V},\mathcal{E},W\}$, the betweenness centrality for the edge $e\in\mathcal{E}$ is defined as $C_{B_e}(e)=\sum_{u\neq v\in\mathcal{V}}\sigma_{uv}(e)/\sigma_{uv}$, where σ_{uv} the number of shortest paths connecting u to v, and $\sigma_{uv}(e)$ the number of shortest paths connecting u to v passing through the edge e. We then arrange edges in ascending order of their betweenness $\{C_{B_e}(e)_i\}$, $i=1,2,\ldots,|\mathcal{E}|$ as $C_{B_e}(e)_{(1)}\leq C_{B_e}(e)_{(2)}\leq\cdots\leq C_{B_e}(e)_{(|\mathcal{E}|)}$. Here $C_{B_e}(e)_{(i)}$ is said to be the ith-order edge betweenness score, or the ith-edge betweenness order statistic.*

Note that the Girvan-Newman algorithm on edge betweenness infers the edges connecting communities, that is, the edges exhibiting a more profound role in the network organization. As a result, P-DropEdge offers multi-fold benefits: (i) it constrains direction of a random walk and acts as a "self-avoiding" random walk, e.g., reduces the chance of moving back to the already visited graph structure; (ii) increases variability among randomly deformed copies of the original graph. That is, let us consider, e.g., an international political network. Randomly removing connections among Mr. and Mrs. Smith or even US Senators from Texas and California will tend to deliver a more similar resulting graph structure than randomly removing collaboration links between Trump, Macron, Putin and Johnson.

Algorithm 1 P-DropEdge Algorithm

Input: Data adjacency matrix W, **Parameter** $p_{\text{P-D.E.}}$, τ
for $i=1$ **to** $|\mathcal{E}|$ **do**
  calculate $C_{B_e}(e)_i$ for the edge e_i
end for
▷ Find order statistics of $\{C_{B_e}(e)_i\}$, $i=1,2,\ldots,\mathcal{E}$
  $C_{B_e}(e)_{(1)}\leq C_{B_e}(e)_{(2)}\leq\cdots\leq C_{B_e}(e)_{(|\mathcal{E}|)}$
▷ Draw edges which correspond to the top order statistics
  $C_{B_e}(e)_{((1-\tau)|\mathcal{E}|)}\leq\cdots\leq C_{B_e}(e)_{(|\mathcal{E}|)}$
▷ Assign weights to the selected edges in 1st-round as
$\psi_{(j)}=\frac{C_{B_e}(e)_{(j)}}{\sum_{j=(1-\tau)|\mathcal{E}|}^{|\mathcal{E}|}C_{B_e}(e)_{(j)}}$, $j=(1-\tau)|\mathcal{E}|,\ldots,|\mathcal{E}|$,
  $\boldsymbol{\psi}=\{\psi_{((1-\tau)|\mathcal{E}|)},\cdots,\psi_{(|\mathcal{E}|)}\}$
▷ Weighted sampling without replacement
  randsample$((1-\tau)|\mathcal{E}|:|\mathcal{E}|,p_{\text{P-D.E.}}\times\tau\times|\mathcal{E}|,\boldsymbol{\psi})$,
  obtain sample $\boldsymbol{s}$ with size $p_{\text{P-D.E.}}\times\tau\times|\mathcal{E}|$
Output new adjacency matrix $W_{p_{\text{P-D.E.}}}=W-W_{C_{B_e}(e)\in\boldsymbol{s}}$

P-DropEdge method is presented in Algorithm 1. Given a resulting adjacency matrix $W_{p_{\text{P-D.E.}}}$ upon P-DropEdge

application, a new fractional Laplacian takes the form $\tilde{L}_{p_{\text{P-D.E.}}} = D^{-\sigma}_{\gamma_{p_{\text{P-D.E.}}}} W_{\gamma_{p_{\text{P-D.E.}}}} D^{\sigma-1}_{\gamma_{p_{\text{P-D.E.}}}}$. Finally, we replace the fractional Laplacian $\tilde{L}$ in (7) with $\tilde{L}_{p_{\text{P-D.E.}}}$ for propagation and training. In validation and testing steps, P-DropEdge is not utilized.

Advantages of LFGCN vs. Higher-order methods Recently there has been a spike of interests to higher-order methods, that is, algorithms based on the graph convolutional layer with higher-order information in graphs, such as APPNP [22], VPN [23], and MixHop [24]. In contrast to such higher-order graph architectures, LFGCN offers multi-fold benefits: (i) Due to a fractal character, LF integrates local graph exploration with long-range excursions, which reduces oversampling comparing to standard random walks and allows for more efficient graph exploration; (ii) since high-order schemes [23, 24] are based on integer powers of Laplacian, when exploring 2-hops, 3-hops,..., k-hops, standard random walks employed in these higher-order methods can only describe larger scale graph structures, often resulting in very dense adjacency matrices and higher computation costs; (iii) Lévy Flights average return probability is lower than average return probability of a normal random walk, implying more efficient graph exploration; (iv) Lévy Flights reinforces separability of clusters, enhances performance for unbalanced data, and is known to yield better search results in directional data. Clearly, these advantages are essential for learning graphs with higher heterogeneity, and for more homogeneous and balanced graphs, methods based on standard random walks may be a competitive alternative.

IV. Experimental Settings

Directed and Undirected Datasets Joining the previous works practice, we use three undirected citation networks benchmark datasets for semi-supervised learning evaluation, including Cora-ML (this Cora dataset consists of Machine Learning papers), CiteSeer and PubMed. We also evaluate our method on four directed networks – Cora, IEEE 118-bus system (IEEE bus), Texas 2000-bus system (TX bus), and South Carolina 500-bus system (SC bus). The dataset statistics are summarized in Table V (in Appendix X). We provide the more details about datasets description on Github in the Appendix X.

Baseline Methods On undirected networks, we compare LFGCN with the following state-of-the-art semi-supervised classification approaches which include (i) using the label matrix as input: label propagation (LP) [34]; (ii) using the feature matrix as input: DeepWalk (DW) [35], graph attention networks (GAT), GNN with ChebNet polynomials filter (ChebNet), GCN, GNNs with convolutional ARMA filters (ARMA), Graph Markov Neural Networks (GMNN) [12], Large-Scale Learnable Graph Convolutional Networks (LGCNs) [36], Shortest Path Graph Attention Network (SPAGAN) [37], APPNP, VPN, and MixHop. In addition, on undirected graphs, we use MotifNet, ChebNet, GCN, ARMA, GMNN, LGCNs, SPAGAN, APPNP, VPN, and MixHop as the benchmarks.

Training Settings Training task is done by using Adam optimizer with learning rate $lr_1 = 0.01$ for undirected networks and $lr_2 = \{0.1; 0.001\}$ for directed networks. To prevent our approach from over-fitting, we consider both adding dropout layer before two graph convolutional layers and kernel regularizers (ℓ_2) in each layer. For undirected and directed networks: we follow the same experimental setup used in the baselines experiments to set the parameters of baselines. Parameters $p_{\text{P-D.E.}}$ and τ largely depend on the distributional properties of a network and can be estimated, e.g., via cross-validation. As a rule of thumb, we recommend $p_{\text{P-D.E.}}$ and τ of 5% and 6%, respectively, in larger networks of more than 2,000 nodes, and $p_{\text{P-D.E.}}$ and τ of 1% and 2%, respectively, in smaller network of less than 1,000 nodes. The best hyperparameter configurations of LFGCN for each dataset by using standard grid search mechanism are available at Github link in Appendix X.

V. Results

1. Performance analysis Tables I and II report the average accuracy delivered by LFGCN and competing methods for undirected and directed networks, respectively. The best performance for each dataset is marked in bold. We find that LFGCN outperforms all competing approaches in all datasets, except for PubMed (LFGCN delivers the second best accuracy result). The improvement gain of LFGCN over the next most accurate method ranges from 0.29% (for CiteSeer over GMNN) to 4.27% (for directed IEEE 118-Bus over GMNN). Remarkably, methods that are applicable both to undirected and directed networks (i.e., [5, 6, 8, 12, 22, 23, 24, 36, 37]) tend to deliver noticeably lower accuracy results for a directed networks (especially on weighted-directed networks), while the new LFGCN method yields a more stable performance across both directed and undirected networks. In turn, PubMed (unweighted-undirected), GMNN outperforms LFGCN up to 2.63%. Based on the obtain results, the new LFGCN approach tends to be the most competitive and, hence, preferred node classification method for sparser networks with higher label rates. Furthermore, the IEEE 118-Bus dataset is the smallest among the considered data, and we might expect to observe lower accuracy results for this dataset due to a limited training set. However, the accuracy yielded by LFGCN is among the highest ones across all datasets. For PubMed, it has the lightest tails for the degree distribution and a weak structural info (i.e., with very few links per node on average), thus LFGCN is not the best exploration choice.

We provide the training time per epoch on all datasets in the Appendix X (see Tables VI VII and VIII).

2. Ablation study by removing individual components in LFGCN To discover the vital components in the success of our LFGCN, we investigate the contributions of individual components proposed in Section III-B to the performance of LFGCN. We conduct experiments by removing individual component separately (in the spirit of leave-one-out operation) from our LFGCN architecture, leading to a network without P-DropEdge, parallel structure, residual block, or *gated max-average pooling*. Table III provides the comparison results between LFGCN without P-DropEdge, parallel structure, residual block, or *gated max-average pooling*. The results show that LFGCN consistently outperforms the reduced LFGCN baselines by a significant margin, reaching around 0.24% to 2.91% relative improvement on Cora-ML and IEEE 118-bus system. These results demonstrate contributions of all components to performance improvement.

3. P-DropEdge vs. DropEdge We now evaluate our P-DropEdge and regular DropEdge of [10] based on the LFGCN and GMNN (i.e., the best performing baseline).

Table I: Comparison of average accuracy (%) and standard deviation (%) in () of semi-supervised classification approaches for undirected networks.

Method	Cora-ML	CiteSeer	PubMed
LP [38]	68.70	46.32	65.92
DW [35]	67.20	43.27	65.33
ChebNet [5]	81.45	70.23	78.40
GCN [6]	81.50	71.11	79.00
ARMA [8]	82.80 (0.63)	72.30 (0.44)	78.80 (0.30)
GAT [17]	83.11 (0.70)	70.85 (0.70)	78.56 (0.31)
GMNN [12]	83.72 (0.90)	73.10 (0.79)	**81.80 (0.53)**
LGCNs [36]	83.35 (0.51)	73.08 (0.63)	79.51 (0.22)
SPAGAN [37]	83.63 (0.55)	73.02 (0.41)	79.60 (0.40)
APPNP [22]	83.31 (0.53)	72.30 (0.51)	80.12 (0.20)
VPN [23]	81.89 (0.57)	71.40 (0.32)	79.60 (0.39)
MixHop [24]	81.90 (0.81)	71.41 (0.40)	80.81 (0.58)
LFGCN w/o $p_{\text{P-D.E.}}$	84.35 (0.57)	71.89 (0.77)	79.60 (0.55)
LFGCN	**84.63 (0.55)**	**73.31 (0.76)**	79.65 (0.50)

Table II: Comparison of average accuracy (%) and standard deviation (%) in () of semi-supervised classification approaches for directed networks.

Method	Cora	IEEE Bus	TX Bus	SC Bus
MotifNet [25]	60.00	65.75	82.00	95.18
ChebNet [5]	58.93	60.00	80.04	94.13
GCN [6]	57.75	52.86	73.36	90.33
ARMA [8]	58.99 (0.52)	70.55 (2.23)	81.20 (0.23)	94.33 (0.47)
GMNN [12]	61.20 (0.50)	78.88 (2.50)	86.21 (0.29)	96.57 (0.52)
LGCNs [36]	60.72 (0.43)	71.43 (2.20)	85.57 (0.25)	95.14 (0.45)
SPAGAN [37]	61.00 (0.45)	78.55 (2.25)	86.00 (0.26)	96.19 (0.40)
APPNP [22]	61.00 (0.44)	82.05 (2.24)	86.77 (0.30)	96.11 (0.45)
VPN [23]	60.53 (0.43)	82.19 (2.20)	86.87 (0.33)	96.23 (0.50)
MixHop [24]	60.33 (0.55)	80.05 (2.50)	86.10 (0.25)	95.94 (0.40)
LFGCN w/o $p_{\text{P-D.E.}}$	60.70 (0.47)	82.20 (2.30)	87.74 (0.30)	97.61 (0.47)
LFGCN	**61.35 (0.45)**	**82.40 (2.24)**	**88.23 (0.26)**	**97.85 (0.47)**

Table IV presents comparison between LFGCN and GMNN with regular DropEdge and P-DropEdge on CiteSeer and South Carolina 500-bus system. We find that while a sufficiently sampling-based edge-removing is helpful for performance enhancement, regular randomly removing edges do not always improve performance. Note that this finding is in contrast to the regular DropEdge where both LFGCN and baseline equipped with P-DropEdge achieve consistently better performance than others. These findings prove the effectiveness of employing preferential approach of P-DropEdge before the learning task.

4. Evaluation of LFGCN-specific parameters During grid search over three parameters (i.e., α, σ, and γ), we find that: (i) the regularization parameter α which used to specify the relative importance of a graph in clustering strongly relates to the probability of initial conditions for random walks when the self-refreshing process works, and it strongly influences the network's generalization ability and node classification performance for all datasets; (ii) the free unifying parameter σ provides enough flexibility to construct a canonical formulation of different graph-based semi-supervised methods – Table I and Table II indicate that the optimal value of σ depends on both the types of networks (undirected and directed) and label rate not on the size of network; (iii) the fractional power parameter γ substantially impacts the accuracy of node classification for the small datasets (see e.g., Figure 2), however, no similarly strong influence is found in the larger datasets.

Table III: Comparison of the LFGCN with/without (1) P-DropEdge ($p_{\text{P-D.E.}}$), (2) Parallel Structure (PS), (3) Residual Block (RB), and (4) *gated max-average pooling* (Gated Max-Avg) in terms of node classification accuracy (%) on undirected Cora-ML and directed IEEE118 bus system. Numbers in () are relative gains in (%) between the best result delivered by the LFGCN and reduced LFGCN.

	Undirected	Directed
Method	Cora-ML	IEEE Bus
LFGCN w/o $p_{\text{G-N.}}$	84.35 (0.33)	82.20 (0.24)
LFGCN w/o PS	84.31 (0.38)	80.71 (2.09)
LFGCN w/o RB	83.31 (1.58)	80.07 (2.91)
LFGCN w/o Gated Max-Avg	83.66 (1.16)	81.12 (1.58)
LFGCN	**84.63**	**82.40**

Table IV: Comparison of GMNN and LFGCN with regular DropEdge (p) and P-DropEdge ($p_{\text{P-D.E.}}$) in terms of node classification accuracy (%) on undirected Citeseer and directed South Carolina bus system. The numbers in () denote the optimal edge removal rate for the models with DropEdge and P-DropEdge.

	Undirected	Directed
Method	CiteSeer	SC Bus
GMNN with p	73.10 (5%)	96.57 (1%)
GMNN with $p_{\text{P-D.E.}}$	73.20 (5%)	96.89 (1%)
LFGCN with p	72.05 (5%)	97.58 (1%)
LFGCN with $p_{\text{P-D.E.}}$	**73.31** (5%)	**97.85** (1%)

5. Hyperparameter sensitivity In the sensitivity analysis setting, we have the ability to analyze the sensitivity of the node classification accuracy to variation from three LFGCN-specific parameters – $\alpha \in \{0.1,\cdots,1\}$, $\sigma \in \{0,0.1,\cdots,1\}$, and $\gamma \in \{0.001,0.01,0.1,1\}$. In this case, we only show the results from sensitively analysis for LFGCN model on IEEE 118-bus dataset. First, we perform the parameter learning experiments on four scenarios with a fixed parameter γ. Figure 2 shows that the accuracy substantially decreases when α is larger than 0.8, especially in γ equals to 0.001 and 0.01 (see Figure 2(a), 2(b)). Setting $\gamma = \{0.1,1\}$, we observe that the classification accuracy nearly monotonic decreases while increasing α. Additionally, LFGCN generally gives consistent and higher accuracy for $\gamma = \{0.001,0.01\}$ when the α parameter is within the range of $\{0.1, 0.2, 0.3, 0.4\}$. We then explore the variation of accuracy based on tuning parameter γ within the range of $[0.001,0.002,\cdots,0.01]$ (setting $\sigma \in \{0,0.1,\cdots,1\}$ at the same time), however, it is hard to obtain the optimal $(\hat{\sigma},\hat{\gamma})$ combination through gathering finite experimental results (100 runs) since some of the results are very close. Therefore, we run the following experiments to demonstrate the impact evaluation of γ:

- *Step 1:* Set $\gamma \in \{0.001,0.002,\cdots,0.1\}$.
- *Step 2:* For each fixed γ, we run our proposed model 100 times separately for σ from 0 to 1 by 0.01. We obtain 100 average accuracies for each γ: $\{\text{Acc}_{\sigma=0},\text{Acc}_{\sigma=0.01},\cdots,\text{Acc}_{\sigma=1}\}^{[i]}$, $1 \le i \le 10$.
- *Step 3:* Fit the Gaussian distribution to $\{\text{Acc}_{\sigma=0},\text{Acc}_{\sigma=0.01},\cdots,\text{Acc}_{\sigma=1}\}^{[i]}$ (see Fig. 3(b)).

Similar to γ, we fit the Gaussian distribution to $\{\text{Acc}_{\gamma=0.001}, \text{Acc}_{\gamma=0.002}, \cdots, \text{Acc}_{\gamma=0.01}\}^{[j]}$ by fixing σ,

where $j = 1, \cdots, 11$ (see Figure 3(a)).

Figure 3 shows that there exists a more profound difference between the shapes of approximate Gaussian distributions by fixing the parameter σ than fixing the parameter γ. These findings imply that σ tends to be a more important factor in the LFGCN approach for small datasets.

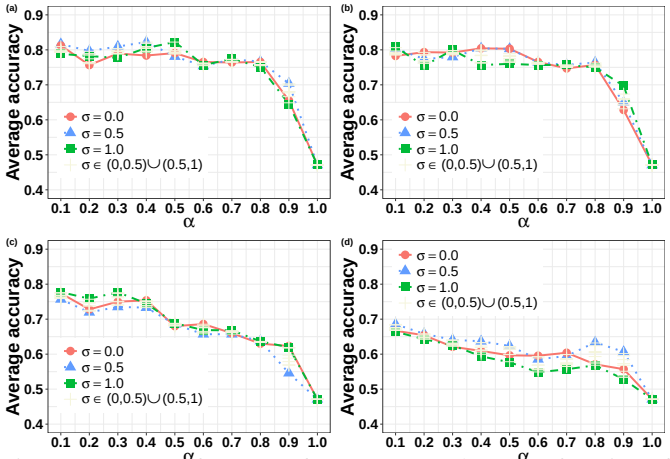

Fig. 2: Accuracy of LFGCN for PR, NL, and SL as a function of α and fractional power $\gamma = \{0.001$ (a), 0.01 (b), 0.1 (c), 1.0 (d)$\}$.

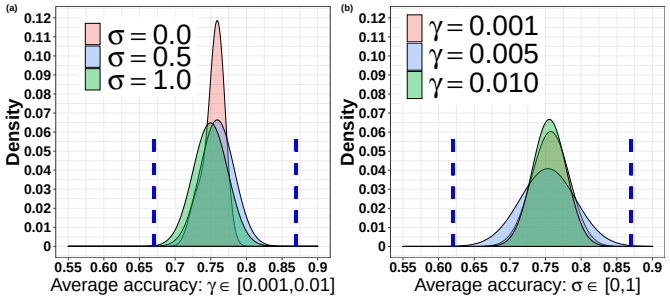

Fig. 3: Generalized Gaussian density of accuracy of LFGCN (two blue dashed lines represent lower and upper bounds, respectively): (a) the red filled curve is the PR-based method ($\sigma = 0$), blue filled curve is the NL-based method ($\sigma = 0.5$), and green filled curve is the SL-based method ($\sigma = 1$). (b) the red, blue, green filled curves represent scenarios with fractional parameters γ of 0.001, 0.005, and 0.010, respectively.

VI. Conclusion

We have proposed a new Lévy Flights Graph Convolutional Network (LFGCN) method for semi-supervised learning on graphs that enables to better capture the intrinsic local graph topology. In addition, to further mitigate over-fitting and over-smoothing, we have proposed a new preferential P-DropEdge algorithm, based on censoring higher edge betweenness order statistics. We have investigated theoretical properties of LFGCN and have validated utility of individual components of the LFGCN architecture.

Our numerical studies have indicated that the new LFGCN method tends to outperform all competing deep learning approaches on both unweighted-directed and unweighted-undirected graphs in all considered datasets, except of PubMed. The gain in learning accuracy of LFGCN over the next best competitor ranges from 0.29% to 4.27%, and the highest gain has been achieved for the IEEE 118-Bus dataset which is the smallest among the considered datasets. Furthermore, in contrast to the competing approaches, LFGCN tends to deliver a more stable performance across directed and undirected networks regardless of the label rate.

In the future we plan to advance the proposed LFGCN technique to learning on multilayer networks, explore utility of P-DropEdge combined with other order statistics on graphs, and enhance graph learning process with topological information on the underlying deep neural network.

References

[1] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[2] 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 *CVRPR*, 2017, pp. 5115–5124.

[3] F. Monti and et al., "Geometric deep learning," 2019, http://geometricdeeplearning.com/.

[4] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[5] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016, pp. 3844–3852.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[7] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

[8] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Graph neural networks with convolutional arma filters," *arXiv preprint arXiv:1901.01343*, 2019.

[9] A. P. Riascos and J. L. Mateos, "Long-range navigation on complex networks using lévy random walks," *Physical Rev. E*, vol. 86, no. 5, p. 056110, 2012.

[10] Y. Rong, W. Huang, T. Xu, and J. Huang, "The truly deep graph convolutional networks for node classification," *arXiv preprint arXiv:1907.10903*, 2019.

[11] N. Balakrishnan and E. Cramer, "The art of progressive censoring," *Statistics for Industry and Technology*, 2014.

[12] M. Qu, Y. Bengio, and J. Tang, "Gmnn: Graph markov neural networks," *arXiv preprint arXiv:1905.06214*, 2019.

[13] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures on AI and ML*, vol. 3, no. 1, pp. 1–130, 2009.

[14] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *NIPS*, 2004, pp. 321–328.

[15] D. Zhou and C. J. Burges, "Spectral clustering and transductive learning with multiple views," in *ICML*, 2007, pp. 1159–1166.
[16] K. Avrachenkov, A. Mishenin, P. Gonçalves, and M. Sokol, "Generalized optimization framework for graph-based semi-supervised learning," in *SIAM SDM*, 2012, pp. 966–974.
[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.
[18] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *NIPS*, 2018, pp. 362–373.
[19] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *WWW*, 2018, pp. 499–508.
[20] J. Chen, T. Ma, and C. Xiao, "Fastgcn: fast learning with graph convolutional networks via importance sampling," *arXiv preprint arXiv:1801.10247*, 2018.
[21] F. Wu, T. Zhang, A. H. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," *arXiv preprint arXiv:1902.07153*, 2019.
[22] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," *arXiv preprint arXiv:1810.05997*, 2018.
[23] M. Jin, H. Chang, W. Zhu, and S. Sojoudi, "Power up! robust graph convolutional network based on graph powering," 2019.
[24] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *ICML*, 2019.
[25] F. Monti, K. Otness, and M. M. Bronstein, "Motifnet: a motif-based graph convolutional network for directed graphs," in *2018 IEEE Data Science Workshop (DSW)*. IEEE, 2018, pp. 225–228.
[26] S. De Nigris, E. Bautista, P. Abry, K. Avrachenkov, and P. Gonçalves, "Fractional graph-based semi-supervised learning," in *25th EUSIPCO*, 2017, pp. 356–360.
[27] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in *AISTATS*, 2016, pp. 464–472.
[28] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1024–1034.
[29] Z. Liu, C. Chen, L. Li, J. Zhou, X. Li, L. Song, and Y. Qi, "Geniepath: Graph neural networks with adaptive receptive paths," in *AAAI*, vol. 33, 2019, pp. 4424–4431.
[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.
[31] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.
[32] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *PNAS*, vol. 99, no. 12, pp. 7821–7826, 2002.
[33] N. Balakrishnan and E. Cramer, "Progressive censoring from heterogeneous distributions with applications to robustness," *Annals of the Institute of Statistical Mathematics*, vol. 60, no. 1, pp. 151–171, 2008.
[34] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," 2002.
[35] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *ACM SIGKDD*, 2014, pp. 701–710.
[36] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *ACM SIGKDD*, 2018, pp. 1416–1424.
[37] Y. Yang, X. Wang, M. Song, J. Yuan, and D. Tao, "Spagan: shortest path graph attention network," in *IJCAI*, 2019, pp. 4099–4105.
[38] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, 2003, pp. 912–919.
[39] E. J. Henley and H. Kumamoto, *Reliability engineering and risk assessment*. Prentice-Hall Englewood Cliffs (NJ), 1981, vol. 568.
[40] C.-K. Wong and M. C. Easton, "An efficient method for weighted sampling without replacement," *SIAM Journal on Computing*, vol. 9, no. 1, pp. 111–113, 1980.
[41] C. T. Scott and M. Köhl, "Sampling with partial replacement and stratification," *Forest Science*, vol. 40, no. 1, pp. 30–46, 1994.
[42] O. Ozturk and N. Balakrishnan, "Constructing quantile confidence intervals using extended simple random sample in finite populations," *Statistics*, pp. 1–15, 2019.
[43] A. S. Nowak and K. R. Collins, *Reliability of structures*. CRC Press, 2012.

APPENDIX

VII. PROOF OF LEMMA 1

Proof. The objective function of the generalized semi-supervised learning framework (i.e., Lemma 1) can be rewritten in the following matrix form: $Q(F) = 2\sum_{k=1}^{K} F_{.k}^T D^{\sigma-1} L D^{\sigma-1} F_{.k} + \mu \sum_{k=1}^{K} (F_{.k}-Y_{.k})^T D^{2\sigma-1} (F_{.k}-Y_{.k})$. Given the first order optimality condition $D_{F.k}Q(F) = 0$, we have $2F_{.k}^T \left(D^{\sigma-1}LD^{\sigma-1}+D^{\sigma-1}L^T D^{\sigma-1}\right) + 2\mu (F_{.k}-Y_{.k})^T D^{2\sigma-1} = 0$. Multiplying the above expression from the right hand side by $D^{-2\sigma+1}$ leads to $2F_{.k}^T \left(D^{\sigma-1}\left(L+L^T\right)D^{-\sigma}\right) + 2\mu (F_{.k}-Y_{.k})^T = 0$. Then, substituting $L = D - W$ and rearranging the terms yields $F_{.k}^T \left(2I - D^{\sigma-1}\left(W+W^T\right)D^{-\sigma}+\mu I\right) - \mu Y_{.k}^T = 0$. Since the resulting adjacency matrix W is symmetric, $F_{.k}^T = \mu Y_{.k}^T \left(2I - 2D^{\sigma-1} W D^{-\sigma} + \mu I\right)^{-1}$. □

VIII. PROOF OF LEMMA 2

Proof. Relaxation time of a Markov process is given by the reciprocal of the spectral gap $\min\{\lambda_2, 2-\lambda_n\}$, where $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ are eigenvalues of the normalized Laplacian. From the singular value decomposition $L^\gamma = U^\top \Lambda^\gamma U$ and monotonicity of the power function, we conclude that λ_2^γ and λ_n^γ are the smallest and largest eigenvalues

of the Lévy flight, respectively. Since for $0<\gamma<1$, $x^\gamma>x$ when $0<x<1$ and $x^\gamma<x$ when $1<x<2$, the spectral gap of the Lévy flight increases with respect to the original random walk. Consequently, the relaxation time of the Lévy flight is less than the relaxation time of the original random walk. □

IX. Proof of Lemma 3

Proof. The parallel structure [39] is constructed in a similar spirit as bagging of features in random forest and other ensemble learning methods. The key rationale behind the parallel structure is to reduce variance and increase stability. Let us first consider the first equation in Lemma 2. Suppose that Υ is a random sample drawn without replacement from a finite population $\mathcal{U}=\{1,2,...,N\}$ according to a sampling design $p(\cdot)$. Each unit $i,i=1,...,N$ of $\mathcal{U}$ is associated with $X^{(i)}_{FGS}$, i.e. the i-th output matrix from FGS convolution for new feature matrix X_i. Probability of choosing sample υ is $Pr(\Upsilon=\upsilon)=p(\Upsilon)>0$ for all $\upsilon\in U$ and $\upsilon\neq\emptyset$. Let Z be the indicator variable such that $Z_i=1$ if X_i is in the sample, and 0 otherwise. Hence, probability that the unit $i,i=1,...,N$ is selected, is given by $\pi_i=E(Z_i)$, and probability that units i and j, $i,j=1,...,N$ are selected simultaneously is $\pi_{ij}=E(Z_iZ_j)$; π_i and π_{ij} are called *first order inclusion probability* and *second order inclusion probability*, respectively. Since in the current paper, we consider a simple random sampling design of n units $X^{(i)}_{FGS}$ without replacement from $\mathcal{U}$, $\pi_i=n/N$ and $\pi_{ij}=n(n-1)/N(N-1)$. Let $\bar{X}_{FGS}=\sum_{i\in N}X^{(i)}_{FGS}/n=\sum_{i=1}^{N}Z_iX^{(i)}_{FGS}/n$. Hence, given $\{X^{(1)}_{FGS},...,X^{(N)}_{FGS}\}$, we find that

$$E[\bar{X}_{FGS}]=\sum_{i=1}^{N}E[Z_i]\frac{X^{(i)}_{FGS}}{n}=\sum_{i=1}^{N}\pi_i\frac{X^{(i)}_{FGS}}{n}=\bar{X}^{U}_{FGS},$$

where $\bar{X}_{FGS}$ is be the output matrix P_1 from a pooling layer (see Residual Blocks in main text Figure 1), which implies that $\bar{X}_{FGS}$ is an unbiased estimator of population mean $\bar{X}^{U}_{FGS}$.

In turn,

$$\mathrm{Var}(\bar{X}_{FGS})=\frac{1}{n^2}\left[\sum_{i=1}^{N}(X^{(i)}_{FGS})^2\mathrm{Var}(Z_i)+\sum_{i=1}^{N}\sum_{j\neq i}^{N}X^{(i)}_{FGS}X^{(j)}_{FGS}\mathrm{Cov}(Z_i,Z_j)\right]=\left(1-\frac{n}{N}\right)\frac{S^2}{n},$$

where $S^2=\sum_{i=1}^{N}\left(X^{(i)}_{FGS}-\bar{X}^{U}_{FGS}\right)^2/(N-1)$. Factor $(1-n/N)$ is called the finite population correction (FPC). Hence, the proposed PS structure decreases an original estimation variance S by order of n. Note that we can consider other sampling designs $p(\cdot)$, including, for example, weighted sampling and p-extended simple random sampling with replacement [see discussion by 40, 41, 42, and references therein]. □

The stability result of the parallel structure [43] follows from verbatim application of the probability bound on the intersection of independent events.

X. More Details on Experiments

The statistics of data we used in the experimental section are summarized in Table V. For undirected networks and Cora, we trained and tested our model on the same dataset splits as in [6]; and we use a $10\%/20\%/70\%$ split into training, validation, and test sets for power grid networks.

Table V: Dataset statistics.

Dataset	Vertices	Edges	Features	Classes	Label rate
Cora-ML	2,708	5,429	1,433	7	0.052
CiteSeer	3,327	4,732	3,703	6	0.036
PubMed	19,717	44,338	500	3	0.003
Cora	19,793	65,311	8,710	70	0.100
IEEE Bus	118	182	2	3	0.100
TX Bus	2,000	2,668	5	3	0.100
SC Bus	500	584	5	3	0.100

We report results for the mean training time per epoch of LFGCN for both undirected and directed networks on Tesla V100-SXM2-16GB.

Table VI: Time per epoch for LFGCN training on both undirected and directed networks.

Dataset	Cora-ML	CiteSeer	PubMed	Cora	IEEE	TX	SC
Runtime	0.04s	0.27s	0.22s	0.15s	0.01s	0.14s	0.02s

All high-order approaches, either fractional or integer powers of Laplacian, lead to increased computational costs. Table VII shows running times for 5 SOTAs on Citeseer. LFGCN time is compatible with SOTAs.

Table VII: Runtime on CiteSeer.

Method	GCN	MixHop	VPN	APPNP	LFGCN
Runtime	0.22s	0.41s	0.26s	0.27s	0.27s

To highlight comparison of DropEdge vs. P-DropEdge, we show runtime/per epoch of LFGCN + DropEdge vs. LFGCN + P-DropEdge for Citeseer (edge betweenness is computed offline):

Table VIII: Runtime of LFGCN with two dropedge methods.

Method	LFGCN + DropEdge	LFGCN + P-DropEdge
Runtime	0.28s	0.33s

Note that while calculating edge betweenness may be time consuming, it's performed offline. Different from DropEdge, P-DropEdge is an offline method based on prior edge info and such prior info is not limited to edge betweenness. Instead, we can use, e.g. edge criticality based on percolation theory, k-path edge centrality with time complexity $O(k|E|)$, and centrality based on the edge impact on giant comp – scores particularly important in weighted graphs, e.g power and transportation systems. P-DropEdge can bring new insights on how targeted perturbation of graph topology can assist addressing GCN oversmoothing and GCN sensitivity to attacks.