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*—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)*. GDL is an emerging direction in machine learning which generalizes DL concepts for data in non-Euclidean spaces, e.g., graphs and manifolds, by bridging the gap between graph theory and deep neural networks [1, 2].

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 one 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 one-by-one 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 [3, 4]. 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, contrary to the high-order schemes, e.g., [5, 6], which are based on integer powers of Laplacian and associated standard random walks describing only larger scale graph structures, LFGCN allows for exploring both local and global graph structures with no computational overhead. Forth, Lévy Flights average return probability is lower than average return probability of a normal random walk, implying more efficient graph exploration. Fifth, 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.

To abate over-fitting and over-smoothing in GDL, we also 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 [7] and is rooted in nonparametric methods, specifically, various censoring schemes, for statistical inference on order statistics. In contrast to uniformly removing edges as in DropEdge [7], 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. Intuitively, 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 [8].

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 of 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

One of the first formulations of convolutional neural networks (CNN) based on spectral graph theory is ChebNet [9]. 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 [10] 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) [11], graph convolutional recurrent networks [12], dual graph convolutional networks [13], and simplified version of GCN [14]. By directly powering the graph Laplacian, GCN based on random walks such as approximate personalized propagation of neural predictions (APPNP) [15], variable power network (VPN) [5], and MixHop [6] can learn the relationships between multiple-hops neighborhood.

To extend the success of GCN on undirected graphs to directed graphs, MotifNet of [16] 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 [17] provides more flexible responses than GCN by using parallel and periodic concatenations of the convolutional kernel via the ARMA filter. As a result, ARMA-GCN of [17] 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 [18].

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 W of $\mathcal{G}$, let D be a degree matrix where $d_{ii} = \sum_{j=1}^{N} w_{ij}$ and $L = U^\top \Lambda U$ be a 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 [19] as follows:

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

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 [20], is as follows:

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

The following lemma [21] 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*

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

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

Proof of **Lemma 1** is in Appendix VII (see [22]). The optimization formulation $S(F)$ with the following expression:

$$S(F) = \min_{F}\Big\{2F_{\cdot k}^{T}D^{\sigma-1}LD^{\sigma-1}F_{\cdot k} + \mu(F_{\cdot k}-Y_{\cdot k})^{T}D^{2\sigma-1}(F_{\cdot k}-Y_{\cdot k})\Big\}, \quad (1)$$

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 $F_{\cdot k}$ as close as possible to the labeling function $Y_{\cdot k}$. 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 classification function for $1\leq k\leq K$:

$$F_{\cdot k}=\frac{1-\alpha}{I-\alpha D^{-\sigma}WD^{\sigma-1}}Y_{\cdot k}, \qquad \alpha=2/(2+\mu). \quad (2)$$

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 [23] embeds Lévy Flights into random walks on graphs by constructing from powers of the Laplacian matrix, i.e., the operator L^{γ}. 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, $0<\gamma\leq 1$, 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 for the Lévy Flights $m_{u\to v}^{(\gamma)}$ between any two nodes whose geodesic distance is not infinite is $m_{u\to v}^{(\gamma)}=\delta_{uv}-(L^{\gamma})_{uv}/k_u^{(\gamma)}$, where δ_{uv} is the Kronecker delta, $k_u^{(\gamma)}$ is the fractional degree of the node u and $k_u^{(\gamma)}\equiv(L^{\gamma})_{uu}$. 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 (see [22]). 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. By replacing L with $L^{\gamma}=U^{\top}\Lambda^{\gamma}U$, the new optimization formulation $S^{*}(F)$ yields:

$$S^{*}(F) = \min_{F}\big\{2F_{\cdot k}^{T}D_{\gamma}^{\sigma-1}L^{\gamma}D_{\gamma}^{\sigma-1}F_{\cdot k} + \mu(F_{\cdot k}-Y_{\cdot k})^{T}D_{\gamma}^{2\sigma-1}(F_{\cdot k}-Y_{\cdot k})\big\}, \quad (3)$$

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

Let $0<\gamma\leq 1$, then the closed form solution for (3) can be obtain as: $F_{\cdot k}=(1-\alpha)\left(I-\alpha D_{\gamma}^{-\sigma}W_{\gamma}D_{\gamma}^{\sigma-1}\right)^{-1}Y_{\cdot k}$, for $k=1,...,K$.

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

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: $g_{FGS}(\alpha,\sigma,\gamma)=(1-\alpha)\sum_{i=0}^{\infty}(\alpha\tilde{L})^{i}$, $0<\alpha,\sigma,\gamma\leq 1$. Empirically, we find that $i=\lceil 4\alpha\rceil$ is a sufficient order of approximation. We then obtain the general classification function by multiplying g_{FGS} by the feature matrix X:

$$\bar{\mathcal{X}}=g_{FGS}(\alpha,\sigma,\gamma)X=(1-\alpha)(X^{'})_{i}, \quad (4)$$

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 (4) as a FGS convolutional layer:

$$H^{(t+1)}=\sigma\left((1-\alpha)\sum_{i=0}^{\infty}(\alpha\tilde{L})^{i}H^{(t)}\mathbf{W}^{(t)}\right), \quad (5)$$

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.

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 classification 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 SOTA pooling operation - *gated max-average pooling* [24] 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 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 [25, 26] 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 [27] 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 7], we develop a new preferential DropEdge approach called the *P-DropEdge* which is based on censoring higher edge betweenness order statistics. In particular, DropEdge of [7] uniformly randomly removes a certain proportion of edges from the input graph at each training epoch and as a result 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 [28]. 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 [29]. 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.

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 $C_{B_e}(e)_i=\sum_{u\neq v\in\mathcal{V}}\sigma_{uv}(e)_i/\sigma_{uv}$ for the edge e_i
end for
▷ Find order statistics of $\{C_{B_e}(e)_i\}$, $i=1,2,...,\mathcal{E}$
  $C_{B_e}(e)_{(1)}\leq C_{B_e}(e)_{(2)}\leq\cdots\leq C_{B_e}(e)_{(|\mathcal{E}|)}$
▷ 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
$\psi_{(j)}=\frac{C_{B_e}(e)_{(j)}}{\sum_{j=(1-\tau)|\mathcal{E}|}^{|\mathcal{E}|}C_{B_e}(e)_{(j)}}$, $j=(1-\tau)|\mathcal{E}|,...,|\mathcal{E}|$,
  $\boldsymbol{\psi}=\{\psi_{((1-\tau)|\mathcal{E}|)},\cdots,\psi_{(|\mathcal{E}|)}\}$
▷ 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 (5) with $\tilde{L}_{p_{\text{P-D.E.}}}$ for propagation and training. In validation and testing steps, P-DropEdge is not utilized.

IV. Experimental Settings

Directed and Undirected Datasets Joining the previous works' practice, we use 3 undirected citation networks benchmark datasets Cora-ML, CiteSeer, and PubMed. We also evaluate our method on 4 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 (see [22], Appendix X).

Baseline Methods On undirected networks, we compare LFGCN with the following semi-supervised classification SOTAs: LP [30]; DeepWalk (DW) [31], GAT [11], ChebNet [9], ARMA-GCN [17], GMNN [8], LGCNs [32], SPAGAN [33], APPNP [15], VPN [5], and MixHop [6]. On directed graphs, we use MotifNet, ChebNet, GCN, ARMA-GCN, GMNN, LGCNs, SPAGAN, APPNP, VPN, and MixHop.

Training Settings Training is done by using Adam optimizer with learning rates $lr_1=0.01$ and $lr_2=\{0.1;0.001\}$ for for undirected and directed networks, respectively. To prevent over-fitting, we consider both adding dropout layer before two graph convolutional layers and kernel regularizers (ℓ_2) in each layer. 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

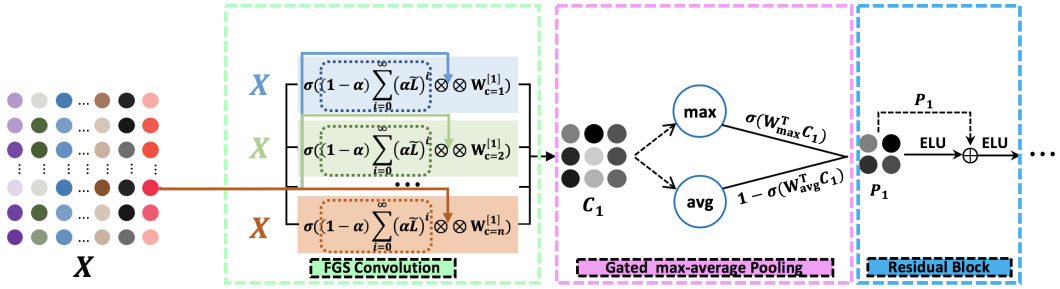

Fig. 1: Illustration Lévy Flights Graph Convolutional Network model. The input is the feature matrix X and the graph within dotted circle represents embedding Lévy Flights into random walks on graph (where L^γ is the Laplacian matrix L to a power γ). LFGCN architecture consists of three main components: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

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.

V. Results

1. Performance analysis Tables I reports the average accuracy delivered by LFGCN and competing methods. 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 2nd 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 bus over GMNN). Remarkably, methods that are applicable both to undirected and directed networks (i.e., [5, 6, 8, 9, 10, 15, 17, 32, 33]) tend to deliver noticeably lower accuracy results for a directed networks (especially on weighted-directed networks), while LFGCN yields a more stable performance across both directed and undirected networks. In turn, in PubMed (unweighted-undirected), GMNN outperforms LFGCN up to 2.63%. Note that PubMed has the lightest tails for the degree distribution and a weak structural info (i.e., with very few links per node on average), hence, LFGCN is not the best exploration choice. These results suggest that LFGCN tends to be the most competitive and, hence, preferred node classification method for sparser networks with higher label rates. Furthermore, the IEEE 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.

2. P-DropEdge vs. DropEdge We now evaluate our P-DropEdge and regular DropEdge of [7] based on the LFGCN and GMNN (i.e., the best performing baseline). Table II presents comparison between LFGCN and GMNN with regular DropEdge and P-DropEdge on CiteSeer and SC 500-bus. 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.

VI. Conclusion

We have proposed a new LFGCN model for semi-supervised learning on graphs that enables to better capture the intrinsic local graph topology. Our studies have indicated that LFGCN tends to outperform all competing DL models on both directed and undirected graphs, especially in sparser regimes.

Acknowledgments

This project was supported in part by NSF DMS 1925346, ECCS 1824716 & IIS 1633331.

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] 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.

[4] T. Michelitsch, A. P. Riascos, B. Collet, A. Nowakowski, and F. Nicolleau, *Fractional Dynamics on Networks and Lattices*. Wiley Online Library, 2019.

[5] M. Jin, H. Chang, W. Zhu, and S. Sojoudi, "Power up! robust graph convolutional network based on graph powering," 2019.

[6] 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.

[7] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *ICML*, 2019.

[8] M. Qu, Y. Bengio, and J. Tang, "Gmnn: Graph markov neural networks," *ICML*, 2019.

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

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

Table I: Average classification accuracy (%) and standard deviation (%) in () for undirected and directed networks.

Method	Cora-ML	CiteSeer	PubMed	Cora	IEEE Bus	TX Bus	SC Bus
LP [34]	68.70	46.32	65.92	-	-	-	-
DW [31]	67.20	43.27	65.33	-	-	-	-
MotifNet [16]	-	-	-	60.00	65.75	82.00	95.18
ChebNet [9]	81.45	70.23	78.40	58.93	60.00	80.04	94.13
GCN [10]	81.50	71.11	79.00	57.75	52.86	73.36	90.33
ARMA-GCN [17]	82.80 (0.63)	72.30 (0.44)	78.80 (0.30)	58.99 (0.52)	70.55 (2.23)	81.20 (0.23)	94.33 (0.47)
GAT [11]	83.11 (0.70)	70.85 (0.70)	78.56 (0.31)	-	-	-	-
GMNN [8]	83.72 (0.90)	73.10 (0.79)	**81.80 (0.53)**	61.20 (0.50)	78.88 (2.50)	86.21 (0.29)	96.57 (0.52)
LGCNs [32]	83.35 (0.51)	73.08 (0.63)	79.51 (0.22)	60.72 (0.43)	71.43 (2.20)	85.57 (0.25)	95.14 (0.45)
SPAGAN [33]	83.63 (0.55)	73.02 (0.41)	79.60 (0.40)	61.00 (0.45)	78.55 (2.25)	86.00 (0.26)	96.19 (0.40)
APPNP [15]	83.31 (0.53)	72.30 (0.51)	80.12 (0.20)	61.00 (0.44)	82.05 (2.24)	86.77 (0.30)	96.11 (0.45)
VPN [5]	81.89 (0.57)	71.40 (0.32)	79.60 (0.39)	60.53 (0.43)	82.19 (2.20)	86.87 (0.33)	96.23 (0.50)
MixHop [6]	81.90 (0.81)	71.41 (0.40)	80.81 (0.58)	60.33 (0.55)	80.05 (2.50)	86.10 (0.25)	95.94 (0.40)
LFGCN w/o $p_{\text{P-D.E.}}$	84.35 (0.57)	71.89 (0.77)	79.60 (0.55)	60.70 (0.47)	82.20 (2.30)	87.74 (0.30)	97.61 (0.47)
LFGCN	**84.63 (0.55)**	**73.31 (0.76)**	79.65 (0.50)	**61.35 (0.45)**	**82.40 (2.24)**	**88.23 (0.26)**	**97.85 (0.47)**

Table II: GMNN and LFGCN with DropEdge (p) vs. P-DropEdge ($p_{\text{P-D.E.}}$); () are the optimal edge removal rates.

	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%)

[11] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *ICLR*, 2018.

[12] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *NIPS*, 2018, pp. 362–373.

[13] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *WWW*, 2018, pp. 499–508.

[14] F. Wu, T. Zhang, A. H. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," *ICML*, 2019.

[15] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," *ICLR*, 2019.

[16] F. Monti, K. Otness, and M. M. Bronstein, "Motifnet: a motif-based graph convolutional network for directed graphs," in *2018 IEEE DSW*, 2018, pp. 225–228.

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

[18] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural. Netw. Learn. Syst.*, 2020.

[19] D. Zhou and C. J. Burges, "Spectral clustering and transductive learning with multiple views," in *ICML*, 2007, pp. 1159–1166.

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

[21] 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.

[22] Y. Chen, Y. R. Gel, and K. Avrachenkov, "Lfgcn: Levitating over graphs with levy flights," *arXiv preprint arXiv:2009.02365*, 2020.

[23] 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.

[24] 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.

[25] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017, pp. 1024–1034.

[26] 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.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[28] 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.

[29] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *PNAS*, vol. 99, no. 12, pp. 7821–7826, 2002.

[30] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," 2002.

[31] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *ACM SIGKDD*, 2014, pp. 701–710.

[32] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *ACM SIGKDD*, 2018, pp. 1416–1424.

[33] Y. Yang, X. Wang, M. Song, J. Yuan, and D. Tao, "Spagan: shortest path graph attention network," in *IJCAI*, 2019, pp. 4099–4105.

[34] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICML*, 2003, pp. 912–919.