A Provable Framework of Learning Graph Embeddings via Summarization

Houquan Zhou[1, 2], **Shenghua Liu**[1, 2*], **Danai Koutra**[3], **Huawei Shen**[1, 2], **Xueqi Cheng**[1, 2]

[1] Institute of Computing Technology, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences
[3] University of Michigan
{zhouhouquan18z, liushenghua, shenhuawei, cxq}@ict.ac.cn, dkoutra@umich.edu.cn

Abstract

Given a large graph, can we learn its node embeddings from a smaller summary graph? What is the relationship between embeddings learned from original graphs and their summary graphs? Graph representation learning plays an important role in many graph mining applications, but learning embeddings of large-scale graphs remains a challenge. Recent works try to alleviate it via graph summarization, which typically includes the three steps: reducing the graph size by combining nodes and edges into supernodes and superedges, learning the supernode embedding on the summary graph and then restoring the embeddings of the original nodes. However, the justification behind those steps is still unknown.

In this work, we propose GELSumm, a well-formulated graph embedding learning framework based on graph summarization, in which we show the theoretical ground of learning from summary graphs and the restoration with the three well-known graph embedding approaches in a closed form. Through extensive experiments on real-world datasets, we demonstrate that our methods can learn graph embeddings with matching or better performance on downstream tasks. This work provides theoretical analysis for learning node embeddings via summarization and helps explain and understand the mechanism of the existing works.

Introduction

Graph representation learning has attracted much research interest in recent years due to its success in various applications including biology, computer vision, text classification, and more. However, learning node representations of large graphs still remains a problem due to high computational complexity and memory usage.

To overcome this problem, some researchers resort to graph summarization methods. They first summarize (some may use the term "coarsening") the input graph into a smaller summary graph by grouping subsets of nodes into supernodes, and then learn supernode embeddings of the summary graph. After that, these embeddings are projected back to the original nodes embeddings with an additional refinement step. Since the size of the graph is decreased, the running time and memory cost are largely reduced. Representative works include HARP (Chen et al. 2018), MILE (Liang, Gurukar, and Parthasarathy 2018), and GraphZoom (Deng et al. 2020).

Method	Kernel matrix	Restoration	Embeddings
$\mathcal{M}$	$\mathcal{K}(\mathcal{G})$	$\mathbf{R}(i,p), v_i \in \mathcal{S}_p$ [1]	$\mathbf{E}$
DeepWalk	$(\mathbf{D}^{-1}\mathbf{A})^{\tau}\mathbf{D}^{-1}$	1	$\mathbf{RE}_s$
LINE	$\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}$	1	
GCN	$\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$	$\sqrt{\frac{d_i}{d_p^{(s)}}}$	
General form	$(\mathbf{D}^{-c}\mathbf{A}\mathbf{D}^{-1+c})^{\tau}$ $\mathbf{D}^{1-2c}$	$\left(\frac{d_i}{d_p^{(s)}}\right)^{1-c}$	

[1] v_i: Node i in original graphs, $\mathcal{S}_p$: Supernode p in summary graphs.

Table 1: The closed-form solutions for learning original graph embeddings from smaller summary graphs by DeepWalk, LINE, and GCN, using restoration matrix $\mathbf{R}$. $\mathbf{E}$ is embeddings on original graph $\mathcal{G}$, and $\mathbf{E}_s$ is embeddings on summary graph $\mathcal{G}_s$.

A key limitation of the existing summarization-based solutions is the lack of theoretical analysis. Existing methods summarize the input graph in a heuristic way and then **empirically** restore the embeddings of the original nodes from the supernode embeddings, without deeply exploring the connection between them. Thus, there are no theoretical studies of the underlying mechanism. Due to the lack of theoretical study, the following questions arise naturally:

- *What is the theoretical mechanism behind learning node embedding via summarization?*
- *What is the connection between node embeddings of original graphs and summary graphs?*
- *Which summarization method should we use, and how should we restore embeddings correspondingly with the* ***theoretical ground?***

In this work, we aim to answer the above questions and fill the theoretical gap. Specifically, we theoretically show that applying three node embedding methods (DeepWalk, LINE, and GCN) on summary graphs is equivalent to applying them on a reconstructed graph based on the configuration model. As a result, original node embeddings can be ap-

*Corresponding Author

proximated by the embeddings of summary graphs through a simple linear transformation. These theoretical results help explain the empirical success of the existing works, and bring more knowledge and understanding about this problem.

Based on the theory, we propose GELSUMM, a novel graph embedding learning framework based on summarization, which aims to learn node embeddings of the input graph from a smaller summary graph, while also providing theoretical justification. We evaluate it on several real-world datasets and show that GELSUMM can learn high quality node embeddings with similar or even better performance efficiently.

In summary, our contributions include:

- **Theory.** We analyze three node embedding learning methods, DeepWalk, LINE and GCN, and theoretically show that embeddings learned on original graphs by them can be approximated by that on summary graphs in a closed-form, as summarized in Table 1.
- **Framework.** Based on the derived theory, we propose a novel and well-formulated framework, GELSUMM, which learns node embeddings for the original graph efficiently from a smaller summary graph.
- **Extensive Experiments.** We conduct thorough empirical analysis on multiple real-world datasets, and demonstrate that our proposed GELSUMM can learn high-quality node embeddings using much less time.

Related Work

Graph representation learning. Graph representation learning (Cai, Zheng, and Chang 2018; Goyal and Ferrara 2018) aims to map each node in graphs into a low-dimensional vector (called embedding or representation) which captures the structural information. The learned latent embeddings can then be fed into machine learning and data mining algorithms for various downstream tasks, such as node classification and link prediction.

A few representative examples from the rich literature in graph representation learning include: DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), node2vec (Grover and Leskovec 2016), LINE (Tang et al. 2015), and graph neural network (GNN) methods (Zhou et al. 2019; Wu et al. 2021), such as GCN (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017) and GAT (Veličković et al. 2018), which adopt a message-passing framework and update the node embeddings based on their neighbors' representations recursively.

Although graph representation learning methods are successful, their lack of scalability and efficiency is an important problem. To tackle this problem, some works employ sampling techniques, including *layer sampling* (Chen, Zhu, and Song 2018; Chen, Ma, and Xiao 2018; Huang et al. 2018) and *subgraph sampling* (Ying et al. 2018; Chiang et al. 2019; Zeng et al. 2020).

Embedding learning by summarization. Another way to improve the scalability of node embedding learning methods is via *graph summarization* (Yan et al. 2019; Liu et al. 2018).

Symbol	Definition
$\mathcal{G}$=$(\mathcal{V}, \mathcal{E})$	Original graph with nodeset $\mathcal{V}$ and edgeset $\mathcal{E}$
$\mathcal{G}_s$=$(\mathcal{V}_s, \mathcal{E}_s)$	Summary graph with supernodes $\mathcal{V}_s$ and superedges $\mathcal{E}_s$
$\mathcal{G}_r$=$(\mathcal{V}, \mathcal{E}_r)$	Reconstructed graph with nodeset $\mathcal{V}$ and edgeset $\mathcal{E}_r$
v_i	Node i in the original graph $\mathcal{G}$
$\mathcal{S}_p$	Supernode p in the summary graph $\mathcal{G}_s$
$d_i, d_p^{(s)}$	Degree of node i and supernode p
$\mathbf{A}, \mathbf{A}_s, \mathbf{A}_r$	Adjacency matrix of original, summary, reconstructed graph
$\mathbf{D}, \mathbf{D}_s$	Degree matrix of original and summary graph
$\mathbf{P}, \mathbf{Q}$	Membership and reconstruction matrix in summarization
$\mathbf{R}$	Restoration matrix for recovering the original embeddings
$\mathbf{E}, \mathbf{E}_s$	Embeddings of original graph and summary graph

Table 2: Major Symbols and Definitions.

The typical approach is to coarsen the original graph into a smaller summary graph, and apply representation learning methods on it to obtain intermediate supernode embeddings. The embeddings of the original nodes are then restored from supernode embeddings with a further refinement step. For example, HARP (Chen et al. 2018) finds a series of smaller graphs which preserve the global structure of the input graph, and learns representations hierarchically. HSRL (Fu, Hou, and Yao 2019) learns embeddings on multi-level summary graphs, and concatenate them to restore original embeddings. MILE (Liang, Gurukar, and Parthasarathy 2018) repeatedly coarsens the input graph into smaller ones using a hybrid matching strategy, and finally refines the embeddings via GCN to obtain the original node embeddings. GPA (Lin et al. 2020) uses METIS (Karypis and Kumar 1998) to partition the graphs, and smooths the restored embeddings via a propagation process. GraphZoom (Deng et al. 2020) employs an extra graph fusion step to combine the structural information and feature information, and then uses a spectral coarsening method to merge nodes based on their spectral similarities. Embeddings are then refined by a graph filter to ensure feature smoothness. (Fahrbach et al. 2020) learns embeddings of the given subset of nodes by coarsening the remaining nodes, which is not capable to learn embeddings of the remaining ones.

These methods, however, are based on heuristic designs and lack theoretical formulation and analysis.

Preliminary

In this section, we introduce two topics, graph representation learning and graph summarization.Table 2 gives the most frequently used symbols in the paper.

Graph Embedding Methods

Within our framework, we theoretically analyze DeepWalk, LINE, and GCN, so we discuss them in more detail here.

DeepWalk and LINE DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) is an unsupervised graph representation learning method inspired by the success of word2vec (Mikolov et al. 2013b,a) in text embedding. It generates random walk sequences and treats them as sentences that are later fed into a skip-gram model with negative sampling to learn latent node representations.

It has been proved in (Qiu et al. 2018) that DeepWalk is implicitly approximating and factorizing the following matrix:

$$\mathbf{M} \coloneqq \log\left(\mathrm{vol}(\mathcal{G})\left(\frac{1}{T}\sum_{\tau=1}^{T}(\mathbf{D}^{-1}\mathbf{A})^{\tau}\right)\mathbf{D}^{-1}\right) - \log b, \quad (1)$$

where T and b are the context window size and the number of negative samples in DeepWalk, respectively.

LINE (Tang et al. 2015) learns embeddings by optimizing a carefully designed objective function that aims to preserve both the first-order and second-order proximity. Though LINE and DeepWalk appear to be different, it has been shown that LINE is also equivalent to factorizing a similar matrix to Eq. (1) and is a special case of DeepWalk for $T = 1$ (Qiu et al. 2018):

$$\mathbf{M} \coloneqq \log\left(\mathrm{vol}(\mathcal{G})\mathbf{D}^{-1}\mathbf{A}\mathbf{D}^{-1}\right) - \log b. \quad (2)$$

GCN GCN (Kipf and Welling 2017) is a graph neural network model transferring traditional convolution neural network to non-Euclidean graph data. In each layer of GCN, node features are propagated based on a first-order approximation of spectral convolutions on graphs:

$$\mathbf{E}^{(k+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{E}^{(k)}\mathbf{W}^{(k)}), \quad (3)$$

where $\mathbf{E}^{(k)}$ are the node embeddings at the k-th layer, $\mathbf{E}^{(0)} = \mathbf{X}$ is the input node feature matrix, $\mathbf{W}^{(k)}$ is a learnable weight matrix at the k-th layer, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the augmented adjacency matrix with self-loops ($\mathbf{I}$ is the identity matrix), and $\tilde{\mathbf{D}}$ is the corresponding augmented degree matrix. Finally, $\sigma(\cdot)$ is the non-linear ReLU operation as an activation function, i.e., $\sigma(x) = \max(0, x)$.

Graph Summarization

Given an input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes, graph summarization aims to find a smaller summary graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ (with $n_s = |\mathcal{V}_s|$ nodes) that preserves the structural information of the original graph. The supernode set $\mathcal{V}_s$ forms a partition of the original node set $\mathcal{V}$ such that every node $v \in \mathcal{V}$ belongs to exactly one supernode $\mathcal{S} \in \mathcal{V}_s$. The supernodes are connected via superedges $\mathcal{E}_s$, which are weighted by the sum of original edges between the constituent nodes. That is, superedge $\mathbf{A}_s(p, q)$ between supernodes $\mathcal{S}_p$, $\mathcal{S}_q$ is defined as:

$$\mathbf{A}_s(p, q) = \sum_{v_i \in \mathcal{S}_p}\sum_{v_j \in \mathcal{S}_q}\mathbf{A}(i, j).$$

The adjacency matrix of the summary graph can be formulated using a *membership matrix* $\mathbf{P} \in \mathbb{R}^{n_s \times n}$ as: $\mathbf{A}_s = \mathbf{P}\mathbf{A}\mathbf{P}^{\mathrm{T}}$, where

$$\mathbf{P}(p, i) = \begin{cases} 1 & \text{if } v_i \in \mathcal{S}_p \\ 0 & \text{otherwise.} \end{cases}$$

Given the summary graph $\mathcal{G}_s$, the original graph $\mathcal{G}$ can be approximated with the reconstructed graphs $\mathcal{G}_r$ with adjacency matrix $\mathbf{A}_r$ defined as:

$$\mathbf{A}_r = \mathbf{Q}\mathbf{A}_s\mathbf{Q}^{\mathrm{T}}, \quad (4)$$

where $\mathbf{Q} \in \mathbb{R}^{n \times n_s}$ is the *reconstruction matrix*. Note that $\mathbf{A}_r$ can be seen as a **low-rank approximation** of the original $\mathbf{A}$.

A natural way to formulate a summarization approach is by minimizing the difference between $\mathbf{A}$ and $\mathbf{A}_r$, so summarization and reconstruction are closely related. Specifically, in this work, we consider the configuration-based reconstruction scheme (Zhou et al. 2021), which adopts *the configuration-based model* as null model. In that case, reconstructed edge weights are proportional to degrees of endpoints. $\mathbf{Q}$ and $\mathbf{A}_r$ are defined as:

$$\mathbf{Q}(i, p) = \begin{cases} \frac{d_i}{d_p^{(s)}} & \text{if } v_i \in \mathcal{S}_p \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

$$\mathbf{A}_r(i, j) = \frac{d_i}{d_p^{(s)}}\mathbf{A}_s(p, q)\frac{d_j}{d_q^{(s)}} \quad v_i \in \mathcal{S}_p, v_j \in \mathcal{S}_q \quad (6)$$

$d_p^{(s)}$ is defined as $d_p^{(s)} = \sum_{v_i \in \mathcal{S}_p} d_i$.

We will later show that, this reconstruction scheme plays an important role in our theoretical analysis.

Proposed methods

Given the problem definition, we now give our theoretical analysis and propose GELSUMM framework.

Theory

In this section, we theoretically reveal the mechanism behind the approach of learning embeddings on summary graphs. In short, we show that running three embedding methods (DeepWalk, LINE and GCN) on a summary graph is equivalent to running them on an approximate configuration-based reconstructed graph.

Approximating kernel matrices We begin our derivation with the following **kernel matrix**, which appears in three node embedding learning methods DeepWalk, LINE and GCN. By comparing the core function of DeepWalk, LINE and GCN, we observe that a common **kernel matrix** can be extracted as:

Definition 1 (Kernel Matrix)**.** *DeepWalk, LINE, and GCN are based on the following generalized kernel matrix:*

$$\mathcal{K}(\mathcal{G}) \coloneqq \left(\mathbf{D}^{-c}\mathbf{A}\mathbf{D}^{-1+c}\right)^{\tau}\mathbf{D}^{1-2c}, \quad (7)$$

where $0 \le c \le 1$ and τ is a positive integer. $\mathbf{A}$ and $\mathbf{D}$ are adjacency matrix and degree matrix of $\mathcal{G}$ respectively.

Remark. *For $c = 1$, we get $\mathcal{K}(\mathcal{G}) = (\mathbf{D}^{-1}\mathbf{A})^{\tau}\mathbf{D}^{-1}$ appears in DeepWalk and LINE (see Eq. (1),(2)). For $c = \frac{1}{2}$, we get $\mathcal{K}(\mathcal{G}) = (\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})^{\tau}$ appears in GCN (see Eq. (3)).*

As we show next in Lemma 1, under **the configuration-based reconstruction scheme** (see Eq. (5) and (6)), this kernel matrix on the original graph, $\mathcal{K}(\mathcal{G})$, can be approximated with the same kernel matrix on the summary graph, $\mathcal{K}(\mathcal{G}_s)$, **in a closed form**.

Lemma 1. *Given $\mathbf{A}_r$ (reconstructed by the configuration-based scheme) as a low-rank approximation of the original adjacency matrix $\mathbf{A}$, the kernel matrix of $\mathcal{G}$ can be approximated by that of $\mathcal{G}_s$:*

$$\begin{aligned}\mathcal{K}(\mathcal{G}) &\approx \left(\mathbf{D}^{-c}\mathbf{A}_r\mathbf{D}^{-1+c}\right)^{\intercal}\mathbf{D}^{1-2c} \\ &= \mathbf{R}\left(\mathbf{D}_s^{-c}\mathbf{A}_s\mathbf{D}_s^{-1+c}\right)^{\intercal}\mathbf{D}_s^{1-2c}\mathbf{R}^{\mathrm{T}} \\ &= \mathbf{R}\ \mathcal{K}(\mathcal{G}_s)\ \mathbf{R}^{\mathrm{T}},\end{aligned} \tag{8}$$

where $\mathbf{R} \in \mathbb{R}^{n\times n_s}$ is the restoration matrix:

$$\mathbf{R}(i,p) = \begin{cases}\left(\frac{d_i}{d_p^{(s)}}\right)^{1-c} & \text{if } v_i \in \mathcal{S}_p \\ 0 & \text{otherwise},\end{cases}$$

which is closely related to the configuration-based reconstruction matrix $\mathbf{Q}$ given in (5)*.*

Proof. See appendix. ■

From this general form of the reconstruction matrix, we obtain specific cases for DeepWalk, LINE, and GCN in the next corollaries.

Corollary 1. *Based on Dfn. 1 of the kernel matrix, $c = 1$ corresponds to DeepWalk and LINE. In this case,* (8) *becomes:*

$$\begin{aligned}\mathcal{K}(\mathcal{G}) &= \left(\mathbf{D}^{-1}\mathbf{A}\right)^{\intercal}\mathbf{D}^{-1} \approx \left(\mathbf{D}^{-1}\mathbf{A}_r\right)^{\intercal}\mathbf{D}^{-1} \\ &= \mathbf{R}\left(\mathbf{D}_s^{-1}\mathbf{A}_s\right)^{\intercal}\mathbf{D}_s^{-1}\mathbf{R}^{\mathrm{T}},\end{aligned} \tag{9}$$

where

$$\mathbf{R}(i,p) = \begin{cases}1 & \text{if } v_i \in \mathcal{S}_p \\ 0 & \text{otherwise}.\end{cases} \tag{10}$$

Corollary 2. *Based on Dfn. 1 of the kernel matrix, $c = \frac{1}{2}$ corresponds to GCN. In this case,* (8) *becomes:*

$$\begin{aligned}\mathcal{K}(\mathcal{G}) &= \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \approx \mathbf{D}^{-\frac{1}{2}}\mathbf{A}_r\mathbf{D}^{-\frac{1}{2}} \\ &= \mathbf{R}\ \left(\mathbf{D}_s^{-\frac{1}{2}}\mathbf{A}_s\mathbf{D}_s^{-\frac{1}{2}}\right)\ \mathbf{R}^{\mathrm{T}},\end{aligned} \tag{11}$$

where

$$\mathbf{R}(i,p) = \begin{cases}\sqrt{\frac{d_i}{d_p^{(s)}}} & \text{if } v_i \in \mathcal{S}_p \\ 0 & \text{otherwise}.\end{cases} \tag{12}$$

Note that $\mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}$ and $\mathbf{R}^{\dagger} = \mathbf{R}^{\mathrm{T}}$ ($\mathbf{R}^{\dagger}$ denotes the pseudo-inverse of $\mathbf{R}$) in Corollary 2, which is important in our analysis of GCN.

Approximating DeepWalk / LINE Based on Corollary 1, we now discuss how to approximate the DeepWalk and LINE node embeddings for the original nodes. Since LINE is a special case of DeepWalk, we focus on the former; similar conclusions can be easily drawn for LINE.

Theorem 1. *Embeddings learned by DeepWalk on the original graph $\mathcal{G}$, $\mathbf{E}$, can be approximated by embeddings learned by DeepWalk on the summary graph $\mathcal{G}_s$, $\mathbf{E}_s$, using the restoration matrix $\mathbf{R}$ in* (10)*, i.e.,*

$$\mathbf{E} \approx \mathbf{R}\ \mathbf{E}_s \tag{13}$$

Proof. See appendix. ■

According to Theorem 1 and the definition of $\mathbf{R}$ matrix ($\mathbf{R}(i,p) = 1$ if $v_i \in \mathcal{S}_p$), we can conclude that nodes in the same supernode get the same embeddings after the restoration. This approach, is exactly the way how related works (including HARP, MILE and GraphZoom) restore the embeddings. Thus, Theorem 1 provides a **theoretical interpretation** for the restoration step of existing methods.

Approximating GCN Given the embeddings $\mathbf{E}_s$ learned by GCN on the summary graph which are usually the output of the last convolution layer, i.e. $\mathbf{E}_s = \mathbf{E}_s^{(K)}$, we can approximate original node embeddings $\mathbf{E}$, as stated in the following theorem.

Theorem 2. *Embeddings learned by GCN on the original graph $\mathcal{G}$ can be approximated by embeddings learned by GCN on the summary graph $\mathcal{G}_s$ with initial features $\mathbf{X}_s \coloneqq \mathbf{R}^{\mathrm{T}}\mathbf{X}$, using the restoration matrix $\mathbf{R}$ defined in* (12)*, in a **least-square approximation** perspective:*

$$\mathbf{E} \approx \mathbf{R}\ \mathbf{E}_s \tag{14}$$

And suppose that the GCN model on the summary graph and the original graph share the same model parameter, then the reconstruction error is bounded by:

$$\|\mathbf{E} - \mathbf{R}\mathbf{E}_s\| \le C^K\sqrt{2\,\mathrm{KL}(\mathbf{A}\|\mathbf{A}_r)}\left(\prod_{l=0}^{K-1}\left\|\mathbf{W}^{(l)}\right\|\right)\left\|\mathbf{E}^{(0)}\right\| \tag{15}$$

where $C = \sqrt{\sum_i(1-\lambda_i)^2}$ is a constant only depends on the input graph. λ_i is the i-th eigenvalue of the normalized Laplacian matrix $\mathbf{L}$ (defined as $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$). And $\mathrm{KL}(\mathbf{A}\|\mathbf{A}_r)$ is the KL-divergence between $\mathbf{A}$ and $\mathbf{A}_r$ (see Appendix), which is included in the objective function of summarization algorithm we use.

Proof. See appendix. ■

Algorithms

From the theory we develop above, learning embeddings on a summary graph with restoration is equivalent to learning embeddings on a graph reconstructed by the configuration model. Thus, the closer the original graph and the reconstructed one is, the better the embeddings are. This fact motivates us to consider the configuration-based reconstruction scheme in the summarization step.

Specifically, we use a graph summarization method DPGS (Zhou et al. 2021) which utilize the configuration-based reconstruction method. It is based on MDL (Minimum Description Length (Rissanen 1978)) principle and aims to minimize the total description length of both the model cost and configuration-based reconstruction error. More details of the summarization algorithm are provided in appendix.

Subsequently, we propose GELSUMM, which aims to learn embeddings for large graphs by summarization. It consists of four steps:

- **(`S1`) Summarization.** First, given the input graph $\mathcal{G}$, we use a configuration-based summarization method(Zhou et al. 2021) to obtain a summary graph $\mathcal{G}_s$.

		DeepWalk				LINE			
		orig	r=0.6	r=0.4	r=0.2	orig	r=0.6	r=0.4	r=0.2
Cora	acc (%)	72.11	73.29	73.87	**74.67**	68.27	66.37	65.63	**68.79**
	time (secs)	107.274	59.452	39.334	**17.255**	14.246	9.313	6.233	**3.172**
Citeseer	acc (%)	46.24	48.87	48.61	**49.08**	41.36	44.81	45.15	**46.13**
	time (secs)	111.022	67.802	42.414	**12.641**	16.991	7.885	5.305	**3.376**
Pubmed	acc (%)	72.35	73.31	73.93	**74.05**	68.51	69.74	**71.40**	69.86
	time (secs)	875.838	523.848	359.602	**175.897**	112.067	65.563	50.331	**31.578**
Flickr	acc (%)	53.19	**53.36**	53.02	52.87	51.47	**52.42**	52.26	50.72
	time (secs)	6142.00	3203.27	2095.68	**1439.87**	528.04	270.05	168.74	**92.12**

Table 3: Node classification results(DeepWalk & LINE). Average running times and accuracy scores over 10 runs are reported. "orig" represents the results on original graphs. r stands for the relative node size. The running time includes the summarization time, the embedding learning time and the refinement time. It can be observed that GELSUMM-DeepWalk and GELSUMM-LINE obtain better results than original DeepWalk and LINE using much less time.

Algorithm 1: GELSUMM-DeepWalk/LINE/GCN

Input: Input graph $\mathcal{G}$, power of filter k, embedding method $\mathcal{M}$
Output: Original embeddings $\mathbf{E}$

1: $\mathcal{G}_s \leftarrow$ Run the configuration-based summarization on $\mathcal{G}$
2: $\mathbf{E}_s \leftarrow \mathcal{M}(\mathcal{G}_s)$
3: $\mathbf{E} \leftarrow \mathbf{R} \cdot \mathbf{E}_s$
4: $\mathbf{E} \leftarrow (\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}})^k \mathbf{E}$
5: **return** $\mathbf{E}$

- **(S2) Embedding Learning.** Next, the chosen embedding learning method $\mathcal{M}$ (DeepWalk, LINE or GCN), is employed on the summary graph $\mathcal{G}_s$ to obtain supernode embeddings $\mathbf{E}_s = \mathcal{M}(\mathcal{G}_s)$.
- **(S3) Restoration.** Then, we restore the embeddings of the original nodes, $\mathbf{E}$, from $\mathbf{E}_s$ using the restoration matrix $\mathbf{R}$, i.e., $\mathbf{E} \leftarrow \mathbf{R}\ \mathbf{E}_s$. The definition of $\mathbf{R}$ varies depending on the method $\mathcal{M}$ used in step (**S2**).
- **(S4) Refinement.** Finally, in order to further refine the restored embeddings $\mathbf{E}$, we apply k times a low-pass smoothing filter, $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ to smooth the restored embedding[1]. It is shown that it can filter out high-frequency noise and keeps low-frequency signals that are useful for downstream tasks (NT and Maehara 2019; Wu et al. 2019; Deng et al. 2020).

Based on the steps described in previous sections and Theorems 1 and 2, we propose three corresponding algorithms: GELSUMM-DeepWalk, GELSUMM-LINE, and GELSUMM-GCN, described in Algorithm 1.

First, a summary graph $\mathcal{G}_s$ is obtained by the configuration-based graph summarization algorithm (**S1**), and then fed into a base embedding learning algorithm to learn embeddings $\mathbf{E}_s$ (**S2**). Original embeddings are then restored from $\mathbf{E}_s$ using corresponding matrix $\mathbf{R}$ (**S3**), and further refined by a smoothing operator (**S4**). The refinement step is efficient and its time cost is ignorable compared to the embedding learning time. Note that in GELSUMM-GCN, self-loops are added to $\mathcal{G}$, since GCN uses augmented adjacency matrix $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$.

To learn supernode embeddings using GCN, we need to assign labels to supernodes. In this paper, we adopt a majority-vote method that assigns the dominating label with the maximum count, i.e., $\text{label}(\mathcal{S}_p) = \arg\max_l |\{v_i | v_i \in \mathcal{S}_p, \text{label}(v_i) = l\}|$. That is, the label of a supernode is the dominating label of its constituent nodes.

[1] $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the augmented adjacency matrix and $\tilde{\mathbf{D}}$ is the corresponding degree matrix

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Type
Cora	2,708	5,278	Citation
Citeseer	3,327	4,552	Citation
Pubmed	19,717	44,324	Citation
Flickr	89,250	899,756	Social
Reddit	232,965	11,606,919	Social
Amazon2M	2,449,029	61,859,076	Co-Purchasing

Table 4: Dataset statistics.

Experiments

In this section, we conduct extensive experiments to answer the following questions:

- **Q1 Effectiveness of GELSUMM-DeepWalk/LINE:** Is GELSUMM-DeepWalk and GELSUMM-LINE able to learn high-quality embeddings efficiently from summary graphs compared to baselines?
- **Q2 Effectiveness of GELSUMM-GCN:** Is GELSUMM-GCN able to learn high-quality embeddings for semi-supervised node classification task?
- **Q3 Scalability:** Is GELSUMM scalable and can be applied to large graphs with up to 2 million nodes?

Experiment Setup. All experiments are conducted on a machine with a 2.40 GHz Intel Xeon E5-2640 CPU, a Tesla K80 GPU, and 128 GB RAM. For the summarization algorithm, we use the source code released by (Zhou et al. 2021). We implement DeepWalk in Python and use LINE's code released by the authors[2].

[2] https://github.com/tangjianpku/LINE

		Cora	Citeseer	Pubmed	Flickr	Reddit
orig	acc(%)	80.20	69.70	78.02	52.89	94.55
	time(secs)	1.92	2.49	5.22	30.77	476.31
$r = 0.6$	GELSumm acc(%)	80.90	70.46	77.38	50.15	93.62
	Empirical acc(%)	80.64	69.94	76.88	48.75	92.55
	time(secs)	1.232	1.4	1.91	22.42	345.72
$r = 0.4$	GELSumm acc(%)	77.76	67.62	76.98	49.80	93.11
	Empirical acc(%)	77.40	66.18	76.6	48.76	87.91
	time(secs)	1.14	1.28	1.65	87.91	292.44
$r = 0.2$	GELSumm acc(%)	75.22	67.64	73.92	49.51	91.90
	Empirical acc(%)	75.24	67.64	73.72	46.15	87.97
	time(secs)	**1.01**	**1.12**	**1.33**	**13.16**	**253.26**

Table 5: Node classification results (GCN) on five datasets. "orig" represents the results on original graphs. Using proposed GELSUMM restoration method yields consistently better results than empirically restoration method.

Datasets. Datasets used in our experiments contain three citation networks, two social networks, and one co-purchasing network. The statistics of them are listed in Table 4. In the first three citation networks, nodes represents documents and edges are citation among them. Bag-of-word vectors are used as node features. For Flickr dataset, each node is an image on the Flickr website, and if two images share some properties, there is an link between them. For Reddit dataset, each node is a post on Reddit, and edges represent co-comment by the same user. The largest dataset Amazon2M, a product network linked by co-purchase behaviors, is tested for scalability. We use the widely-used dataset splits as in (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017).

Q1 Effectiveness of GELSUMM-DeepWalk/LINE

In this experiment, we evaluate GELSUMM-DeepWalk & GELSUMM-LINE on two downstream tasks: node classification and link prediction (placed in appendix due to space limit). Four datasets Cora, Citeseer, Pubmed and Flickr are used in this experiments. Reddit is not included since running representation learning methods on it raises an OOM(out of memory) error on our machine.

Node Classification In node classification task, we summarize the original graph into several summary graphs with different relative sizes (from 0.6 to 0.2), and run DeepWalk/-LINE on summary graphs. The restored embeddings are used to train a Logistic Regression classifier for node classification.

Hyperparameters: Following (Perozzi, Al-Rfou, and Skiena 2014), in DeepWalk, we sample $10 \cdot |\mathcal{V}|$ random walk sequences. The walk length is set to 80 and the window size is set to 10. In LINE, we set the number of negative samples to 5. The embedding dimension is set to 128 for all methods. For GELSUMM-DeepWalk and GELSUMM-LINE, k is set to 2. We give analysis of the effect of paramter k in the appendix.

Results. The experimental results (mean accuracy score and running time of 10 runs) are shown in Table 3. We observe that GELSUMM-DeepWalk and GELSUMM-LINE can match or outperform original DeepWalk and LINE in all datasets. For example, on Citeseer dataset, GELSUMM-DeepWalk get a 2.84% performance increase using $8.78\times$ less time, and GELSUMM-LINE get a 4.77% accuracy increase using $5.03\times$ less time.

Comparison with existing methods. We also compare our GELSUMM framework to the state-of-the-art methods HARP, MILE and GraphZoom. In this experiment, we use node classification task as the base task and choose DeepWalk as the base embedding learning method. For baseline methods, we use the source codes released by the authors off-the-shelf[3]. For MILE and GraphZoom, input networks are summarized at different coarsening levels to produce summary graph with similar sizes. We provide sizes of summary graphs obtained by baseline methods in the appendix. For our GELSUMM, the relative sizes are $[0.6, 0.4, 0.2, 0.1]$ to match the summary sizes.

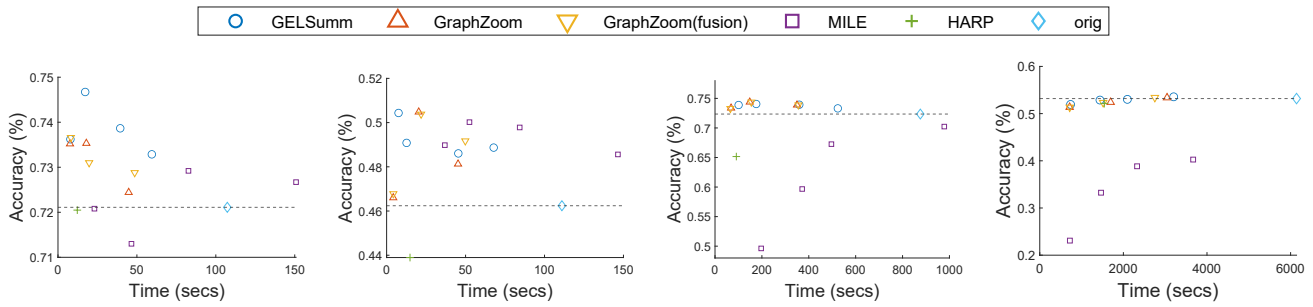
Running time and classification accuracy (average of 10 runs) are displayed in Figure 1. On all datasets, HARP gets worse performance than that on original graphs. MILE keeps relative good performance when the coarsening level is low, but the performance drops when the coarsening level goes higher. Our GELSUMM can obtain promising results with much less time. Compared to GraphZoom, our GELSUMM gets higher performance on Cora dataset, and gets comparable performance using similar time on Citeseer, Pubmed and Flickr datasets. Furthermore, compared to the baselines, our GELSUMM provides more theoretical analysis and interpretation.

Q2 Effectiveness of GELSUMM-GCN:

In this experiment, we test GELSUMM-GCN on semi-supervised node classification task. We train GCN on summary graphs using majority-label assignment. That is, assigning the supernode label as the maximum-count label within it. The restored node embeddings are used for classification after refinement.

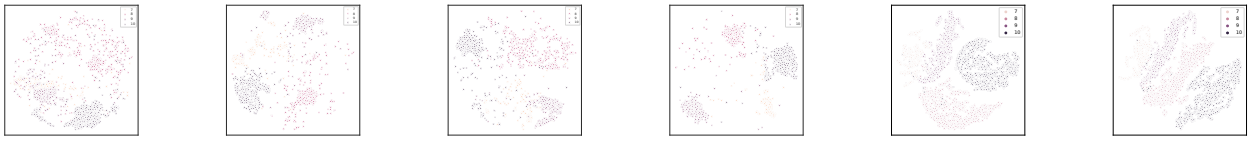
To validate the effect of our theoretically-grounded restoration method, we also report the classification accu-

[3]https://github.com/GTmac/HARP
https://github.com/jiongqian/MILE
https://github.com/cornell-zhang/GraphZoom



(a) Cora (b) Citeseer (c) Pubmed (d) Flickr

Figure 1: Comparison with the baselines. "orig" represents the results on original graphs. Our algorithm obtains promising performance using much less time.



(a) Deepwalk-0.2 accuracy: 0.7441 (b) DeepWalk-0.1 accuracy: 0.7757 (c) LINE-0.2 accuracy: 0.7597 (d) LINE-0.1 accuracy: 0.7586 (e) GCN-0.2 accuracy: 0.8389 (f) GCN-0.1 accuracy: 0.8321

Figure 2: Embeddings of part of nodes from 4 classes learned via GELSUMM on Amazon2M datasets with summarization ratio 0.2 and 0.1. Vertex colors represent its category. Accuracies are given in the captions. Embeddings are projected into 2-dim space via t-SNE. Representations of different classes are well-seperated.

racy using empirical restoration method, that is, assigning the same embedding to nodes within a supernode.

Hyperparameters: For GELSUMM-GCN, the learning rate is set to 0.001 and the dropout rate is set to 0.5. k is set to 1 on all datasets except Flickr and Pubmed (k is set to 2). Training epoch is set to 200 for all datasets.

Results. Experimental results are reported in Table 5. It can be seen that the proposed restoration method (GELSumm acc) obtains consistently better results than empirical restoration method (Empirical acc) adopted by existing works such as MILE and GraphZoom. Compared to original GCN, the accuracy scores drop slightly (less than 4%) after summarization, since the label information is lost during summarization. The time speedup may not look significance since GCN layer costs $O(m)$ and not $O(n)$. However, the performance is still comparable to baselines (within 5%) even when the size of summary graph is only 20% of the size of the original graph.

Q3 Scalability

To test the scalability of our GELSUMM framework, we test on a large Amazon2M dataset, which contains over 2.4 million nodes. Running node embeddings methods on the original graph is not applicable due to memory limit. We apply GELSUMM on summary graphs with summarization ratio 0.2 and 0.1 respectively to get the original embeddings. To show the quality of learned embeddings, we randomly choose some nodes from 4 classes and visualize their embeddings via t-SNE, which is shown in Figure 2. It can be seen that nodes from different classes are well-separated, demonstrating that our GELSUMM can learn matchable embeddings of original nodes in large-scale graphs with limited memory and less time.

Conclusion

In this work, we propose a novel and well-formulated framework, GELSUMM, for learning node embeddings using a summarization approach. By theoretically showing that a specific form of kernel matrix—which appears in DeepWalk, LINE and GCN—can be approximated under a configuration-based reconstruction scheme on the summary graph, we propose three algorithms to learn node embeddings on the summary graph with a closed-form solution. Our study helps understand the empirical success of current related methods, and provides theoretical insights for future works on this problem. Future directions include introducing more embedding methods into our framework, and extend to more graph mining tasks.

Acknowledgments

This paper is partially supported by the National Science Foundation of China under Grant No.91746301, U1911401, U21B2046, 61872206 and the Strategic Priority Research Program of the Chinese Academy of Sciences, Grant No.XDA19020400. This paper is also supported by NSF under CAREER Grant No. IIS 1845491.

References

Cai, H.; Zheng, V. W.; and Chang, K. C.-C. 2018. A comprehensive survey of graph embedding: Problems, techniques,

and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9): 1616–1637.

Chen, H.; Perozzi, B.; Hu, Y.; and Skiena, S. 2018. HARP: Hierarchical Representation Learning for Networks. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press.

Chen, J.; Ma, T.; and Xiao, C. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR*.

Chen, J.; Zhu, J.; and Song, L. 2018. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In *ICML*, 942–950. PMLR.

Chiang, W.-L.; Liu, X.; Si, S.; Li, Y.; Bengio, S.; and Hsieh, C.-J. 2019. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 257–266.

Deng, C.; Zhao, Z.; Wang, Y.; Zhang, Z.; and Feng, Z. 2020. GraphZoom: A Multi-level Spectral Approach for Accurate and Scalable Graph Embedding. In *ICLR*.

Fahrbach, M.; Goranci, G.; Peng, R.; Sachdeva, S.; and Wang, C. 2020. Faster graph embeddings via coarsening. In *ICML*, 2953–2963. PMLR.

Fu, G.; Hou, C.; and Yao, X. 2019. Learning topological representation for networks via hierarchical sampling. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Goyal, P.; and Ferrara, E. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151: 78–94.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Huang, W.; Zhang, T.; Rong, Y.; and Huang, J. 2018. Adaptive sampling towards fast graph representation learning. volume 31.

Karypis, G.; and Kumar, V. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1): 359–392.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Liang, J.; Gurukar, S.; and Parthasarathy, S. 2018. MILE: A Multi-Level Framework for Scalable Graph Embedding. *arXiv:1802.09612*.

Lin, W.; He, F.; Zhang, F.; Cheng, X.; and Cai, H. 2020. Initialization for network embedding: A graph partition approach. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 367–374.

Liu, Y.; Safavi, T.; Dighe, A.; and Koutra, D. 2018. Graph summarization methods and applications: A survey. *ACM Computing Surveys (CSUR)*, 51(3): 1–34.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; and Dean, J. 2013b. Distributed Representations of Words and Phrases and their Compositionality. *arXiv:1310.4546*.

NT, H.; and Maehara, T. 2019. Revisiting Graph Neural Networks: All We Have is Low-Pass Filters. *arXiv:1905.09550*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 701–710. ACM.

Qiu, J.; Dong, Y.; Ma, H.; Li, J.; Wang, K.; and Tang, J. 2018. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 459–467. ACM.

Rissanen, J. 1978. Modeling by shortest data description. *Automatica*, 14(5): 465–471.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. LINE: Large-scale Information Network Embedding. In *WWW*. ACM.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning*, 6861–6871. PMLR.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1): 4–24.

Yan, Y.; Zhu, J.; Duda, M.; Solarz, E.; Sripada, C.; and Koutra, D. 2019. Groupinn: Grouping-based interpretable neural network for classification of limited, noisy brain data. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 772–782.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 974–983.

Zeng, H.; Zhou, H.; Srivastava, A.; Kannan, R.; and Prasanna, V. 2020. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *International Conference on Learning Representations*.

Zhou, H.; Liu, S.; Lee, K.; Shin, K.; Shen, H.; and Cheng, X. 2021. DPGS: degree-preserving graph summarization. 280–288.

Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2019. Graph Neural Networks: A Review of Methods and Applications. *arXiv:1812.08434*.