Privacy Protection and Utility Trade-Off for Social Graph Embedding

Lin Cai[a], Jinchuan Tang[a,*], Shuping Dang[b], Gaojie Chen[c]

[a] *State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang, Guizhou, China*
[b] *Department of Electrical & Electronic Engineering, University of Bristol, Bristol, United Kingdom*
[c] *5G/6G Innovation Centre, Institute for Communication Systems, University of Surrey, Guildford, United Kingdom*

Abstract

In graph embedding protection, deleting the embedding vector of a node does not completely disrupt its structural relationships. The embedding model must be retrained over the network without sensitive nodes, which incurs a waste of computation and offers no protection for ordinary users. Meanwhile, the edge perturbations do not guarantee good utility. This work proposed a new privacy protection and utility trade-off method without retraining. Firstly, since embedding distance reflects the closeness of nodes, we label and group user nodes into sensitive, near-sensitive, and ordinary regions to perform different strengths of privacy protection. The near-sensitive region can reduce the leaking risk of neighbouring nodes connecting to sensitive nodes without sacrificing all of their utility. Secondly, we use mutual information to measure privacy and utility while adapting a single model-based mutual information neural estimator to vector pairs to reduce modeling and computational complexity. Thirdly, by keeping adding different noise to the divided regions and reestimating the mutual information between the original and noise-perturbed embeddings, our framework achieves a good trade-off between privacy and utility. Simulation results show that the proposed framework is superior to state-of-the-art baselines like LPPGE and DPNE.

Keywords: Social networks, graph embedding, mutual information, privacy and

*Corresponding author
Email addresses: gs.cail21@gzu.edu.cn (Lin Cai), jctang@gzu.edu.cn (Jinchuan Tang), shuping.dang@bristol.ac.uk (Shuping Dang), gaojie.chen@surrey.ac.uk (Gaojie Chen)

utility trade-off.

2010 MSC: 00-01, 99-00

---

1. Introduction

With the increasing prevalence of social media platforms like Facebook and Twitter, social networks have become vital for people to communicate and access information daily. Governments and businesses utilize social network data for various purposes including advertisement, healthcare, disease control, disaster prevention, and more [1–3]. Typically, social networks are presented in the form of large-scale adjacency matrices or sparse matrices, and direct analysis requires a vast amount of computation time. The graph embedding can capture structural information of the graph while reducing the amount of computation for downstream analysis. However, graph embedding also aids large-scale privacy leakage where the sensitive structures are extracted from the data by machine learning to infer the private social relationships of key personnel [4]. Besides, ordinary users may be less concerned about privacy, but their connection with others could expose the identity of sensitive users via inference attacks. Especially, the nearer an ordinary node is to a sensitive node, the easier it is to disclose the sensitive node's information. In addition, previous privacy protection methods achieved the effect of privacy protection by constantly perturbing the original graph during the retraining of the embedding process [5]. This, however, would require a largely new model training overhead for a large network, and there was no guarantee for utility performance. Therefore, an efficient privacy protection method for the entire embedding matrix while reducing the training workload is crucial.

Graph embedding technology reduces data dimensionality through mapping into low-dimensional spaces not only simplifies computational tasks but also addresses challenges related to storage complexity [6]. The embedding algorithm automates the learning of feature representations for nodes and graph structures, facilitating an improved expression of similarities and relationships between nodes. The adaptability and generality of graph embedding technology make it an efficient and powerful tool for handling diverse graph data types. Existing graph embedding algorithms can en-

capsulate important information on connections and pathways into a tractable problem with reduced complexity. However, their performance varies. Laplacian Eigenmaps [7] and Locally Linear Embedding (LLE) [8] aim to uncover the underlying structure of high-dimension data by emphasizing the preservation of local relationships. Laplacian Eigenmaps use graph Laplacian matrices and eigenvectors, while LLE focuses on the local linearity of data points in the embedding space. These techniques are valuable for nonlinear dimensionality reduction and visualization of complex datasets. However, scalability remains a significant challenge in these methods. DeepWalk [9] combined natural language processing with unsupervised feature learning in the context of complex networks. It created a random walk sequence for each node and then fed it into the SkipGram model [10] to obtain the embedding feature vector for each node. Due to its online learning approach, DeepWalk could effectively handle large-scale social networks and update the model in real time.

Since embedding low-dimension representation vectors retains graph topology and other related information about the graph in coordinates, protecting them using privacy protection techniques becomes critical. For instance, the author of [11] introduces a differentially private scheme rooted in the dk-graph model for sharing meaningful graph datasets, but the generated graph dataset may not fully retain all statistical properties of the original graph dataset, which may affect the accuracy and effectiveness of some graph analysis tasks. The work in [12] addresses the issue of preserving differential privacy during degree correlation-based graph generation, but these methods do not consider how to balance privacy and utility, thus affecting the quality and usability of the generated graph data. Pioneering works such as [13] and [14] employ differentially private matrix factorization with stochastic gradient descent (SGD) to alleviate accumulated noise. The author of [14] uses two differentially private matrices that are sampled using the exponential mechanism. However, due to the inherent high correlation within graph data, these approaches encounter challenges, especially when applied to matrix factorization-based graph embedding. This is particularly pronounced when the input dataset contains a substantial amount of related data, necessitating the addition of a prohibitive amount of noise. Consequently, the published embedding matrix becomes nearly impractical, failing to maintain the utility of the graph struc-

ture. Moreover, adhering to the sequential composition theory of differential privacy, iterative computations may lead to accumulating errors, resulting in diminished utility for graph embedding. The work in [15] achieves differential privacy in matrix factorization by employing objective perturbation and working together with k-coRating. Meanwhile, researchers from [16] utilize the Frank-Wolfe method [17] with differential privacy as the building block of matrix factorization for tackling matrix completion. Inspired by [13], study [18] introduces differentially private graph embedding via objective function perturbation. However, for bounding the global sensitivity of the target non-private function, they suffer from prohibitively complex analytic calculations, resulting in poor scalability. Additionally, they can only protect one of the two submatrices obtained through matrix factorization, potentially increasing the risk of privacy breaches. Another perspective, as seen in LPPGE [19], employs adversarial learning for link privacy-preserving graph embedding. However, the reconstruction of the graph retains sensitive information about genuine social relationships, posing a potential risk of sensitive information leakage. Although these investigations deal with the differentially private publication of social graph data, none of them specifically consider the preservation of the utility of the embedding matrix. The authors of [20] employ a Shannon entropy improved method to identify edges within the graph, while the algorithm put forth by [21] is proficient in extracting topics, connections, and additional associated resources from the graph, thereby facilitating the creation of interactive dynamic knowledge graphs.

Within social networks, adversaries can leverage publicly available user data on these platforms for training. Some users do not care whether their personal connections are obtained by others but are eager to promote themselves by exposing complete personal data. Attackers capitalize on this auxiliary vulnerability to train their models, subsequently utilizing this data to make inferences about sensitive users' private information [22]. The closer an ordinary user is to a sensitive user, the higher the risk of compromising that user's privacy. Studies such as studies [23, 24] reveal that through various inference attack techniques, it's feasible to infer sensitive users' relevant information based on users' data surrounding the sensitive ones. Membership inference attacks are a common means. The objective of such attacks is to ascertain

whether a specific individual is included in a particular dataset. Through the analysis of model outputs, attackers can understand the relationship between a specific member and particular attributes or behaviors, further revealing the identity of that member. Furthermore, findings from [25] show that deleting nodes and edges can be inferred from the privacy-protected published embedding data. This is because, before node deletion, a graph embedding encoding that satisfies the current relationship is formed. Even after deletion, this relationship still exists to some extent, meaning that deleting sensitive nodes does not remove the sensitive relationship. Therefore, it's possible to reconstruct the original network of relationships between nodes based on this feature, meaning the sensitive network of relationships that sensitive nodes always wanted to protect still exists and does not change due to the deletion of coordinates. Only by deleting sensitive nodes from the original graph can it be ensured that these nodes and their surrounding relationships do not participate in training, thus leading to changes in the topology structure used for training and ensuring that these sensitive relationships are not leaked after training. However, embedding training again after deleting nodes requires re-calling the embedding model, which will lead to expensive retraining for a new embedding model.

To ensure that social network data effectively serves diverse downstream tasks, one must not only consider the effectiveness of privacy protection but also prioritize data utility [26, 27]. Unfortunately, there is currently no privacy protection method that offers an effective trade-off between privacy and utility. Recent works, such as [28], have introduced a privacy funnel utilizing a mutual information neural estimator (MINE) [29] to optimize the balance between privacy and utility through mutual information estimation. This model is characterized by its simplicity, effectiveness, and stability. The studies from [30, 31] demonstrate the possibility of embedding private information into images using steganography, These advancements pose novel challenges for ensuring privacy in graph data. Notably, MINE is capable of calculating mutual information when the data distribution is unknown. Nevertheless, the precise impact and suitability of this approach in graph embedding scenarios require further investigation and assessment. This is not trivial because a graph consists of multiple variables. Training MINE in pairs would require a huge amount of computation for each pair and may require

an independent model. And, the studies on the mathematical properties and stability conditions of neural networks can be supported by the related works in [32] and [33].

It can be observed that previous privacy protection methods often adopt the same privacy protection strength for all nodes. They did not consider that the nodes may have different privacy requirements. Adopting the same privacy protection policy for all nodes will result in insufficient privacy protection for nodes with high sensitivity, and nodes with low sensitivity cannot have the desired data utility. Meanwhile, the studies of [12, 13, 15] employed privacy protection without considering to balance it with the utility. In order to limit the global sensitivity of the target non-private function, the authors of [18] conducted complex analytical calculations with poor scalability. The authors of [14, 16, 17] used the matrix decomposition method to achieve privacy protection by making one of the result matrices satisfy differential privacy. This only protects one of the matrices, which may result in a certain degree of privacy leakage and scalability issues. Additionally, none of these methods provide an effective means of measuring the degree of privacy protection, resulting in a lack of a viable trade-off method.

*Motivation:*This research proposes a new privacy-preserving and utility trade-off method for graph embeddings that does not require retraining embedding, and we take into account the privacy of all users, not just sensitive users. Moreover, our approach addresses the issue of high computational costs in traditional privacy protection methods by achieving a balance between privacy and utility with relatively low time expenses, so that the relationship network of sensitive nodes can be protected while providing balanced utility to downstream tasks.

The contributions of this work are given as follows:

- We opted for a fixed embedding model to process the graph and divided privacy protection based on nodes' embedding closeness to sensitive ones. Subsequently, we proposed a region division algorithm to divide nodes into sensitive, near-sensitive, and ordinary matrices by calculating and sorting in Euclidean distance. It has a small time cost due to its simplicity while still serving the purpose of capturing the closeness of nodes in the embedding space.

- We used mutual information to assess both privacy and utility and proposed a MINE-GE that is suitable for processing the graph embedding vectors, where each element is considered as i.i.d. so that only one model is needed instead of multiple models to calculate mutual information. It reduces the complexity of the model, simplifies the estimation of high-dimensional mutual information, and captures graph relationships.

- Aiming at privacy protection and utility trade-off in graph embedding, we proposed a new privacy and utility trade-off framework. This framework achieves a trade-off between privacy and utility by selecting a fixed embedding matrix without retraining the embedding, dividing different regions, and estimating mutual information. Comparing with existing retraining privacy protection methods, it greatly reduces the time cost of privacy protection operations.

- We substantiated the efficacy of our region division through rigorous experimentation. Compared to conventional baseline techniques, our framework exhibits a distinct superiority in safeguarding the relationship network of sensitive nodes. The results of the membership inference attack and multi-label classification task show that our framework effectively obtains stronger sensitive node privacy and better data utility.

The remainder of our work is organized as follows: Section 2 introduces our system model. Section 3 introduces related techniques. Section 4 introduces the our privacy protection framework in detail. Section 5 provides a detailed account of the simulation results and demonstrates the effectiveness of our method. Finally, Section 6 summarizes the paper and looks toward future work. Notations and Acronyms is shown in Table 1.

Table 1: Notations and Acronyms

Symbol	Meaning
M_n	Embedding matrix
$\hat{M}_n$	Label and group matrix
$\hat{M}_p$	Privacy utility trade-off achieved matrix
$\mathcal{N}_i(0,\sigma^2\mathrm{I})$	Gaussian noise with mean 0 and variance σ^2
ε	Privacy threshold
$G(V,E)$	A graph with V nodes and E edges
$\mathbb{R}$	Real number space
v_i	Low-dimension embedding representation of node i
d	Embedding dimension
$I(\cdot;\cdot)$	Mutual information of two random variables
$\hat{I}_\theta(\hat{\cdot};\hat{\cdot})$	Estimated mutual information
$I_{total}(\cdot;\cdot)$	Total vector mutual information
θ	Deep neural network parameter
k	Mini-batch size
η	Learning rate
$\hat{M}_{sensitive}$	Sensitive nodes sets
$\hat{M}_{near}$	Near-sensitive nodes sets
$\hat{M}_{ordinary}$	Ordinary nodes sets
α	Number of near-sensitive node
$D(\cdot,\cdot)$	Euclidean distance between two vectors
SNR	Signal-to-noise ratio
P_S	Signal power
σ^2	Noise power
a_i	Privacy protection level
Δa_i	Updating steps
a_i^{UB}	Upper bound
a_i^{LB}	Low bound
w	Window length
β	Walks per vertex
t	Walk length
$\mathscr{Y}$	Set of labels
$\lvert\cdot\rvert$	Number of elements in a set
$\lceil\cdot\rceil$	The ceiling function
i.i.d.	Independent and identically distributed
MINE	Mutual information neural estimator
MINE-GE	MINE based on Graph Embedding

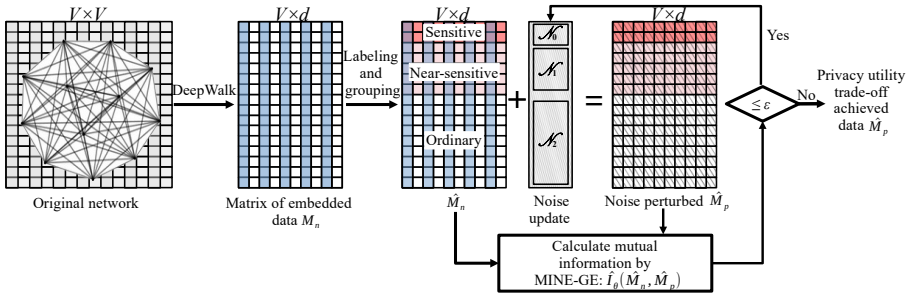


Figure 1: Privacy and utility trade-off system model.

2. System Model

As shown in Fig. 1, we consider that directly publishing embedded matrix data faces the risk of privacy leakage while excessive protection will cause data to fail to serve downstream tasks. Traditional privacy protection methods achieve privacy by deleting nodes. However, in the embedding space, simply removing nodes from the embedding matrix is not effective in safeguarding the relationship network of nodes. To ensure that relevant relationship networks are not leaked, it was necessary to delete corresponding nodes from the original graph and retrain a new model. This is a process that demands significant computational resources. Therefore, it is desirable to have a privacy protection framework to protect privacy and ensure data utility based on embedding results without retraining the embedding model.

According to the characteristics of social networks and the actual situation of each node, the consisting nodes can be divided into sensitive and ordinary regions. However, when the adversary obtains the information of ordinary nodes around the sensitive nodes, it can infer the topology information and relationship information of the sensitive nodes. Therefore, it is necessary to offer stronger protection on these near-sensitive nodes than the rest of the ordinary nodes. We use mutual information to measure the data utility after privacy-preserving operations, but the direct computation of mutual information is difficult. MINE has been proven effective in estimating mutual information, but multiple MINEs are computationally expensive for graph embedding scenarios with multiple variables.

Under the aforementioned considerations, we proposed a privacy-preserving and

utility trade-off framework for graph embedding. As shown in the figure, we employ a random walk based graph embedding method to transform the adjacency matrices into low-dimension vectors, resulting in the creation of the embedding matrix M_n, and we execute the embedding algorithm only once without the need for repetitive embeddings. This would be a huge time saving since one embedding training can take hours for a large network. Euclidean distance due to its simplicity can effectively measure the proximity between two nodes in space. It is used to determine the neighbor nodes around sensitive nodes. We labeled and grouped them into sensitive, near-sensitive, and ordinary nodes according to the distance, obtaining the set $\hat{M}_n$. Subsequently, according to the privacy requirements of different nodes, different levels of Gaussian noise are sampled, and different levels of privacy protection operations are performed on different nodes. Thereby, we can obtain a privacy-protecting embedded node set $\hat{M}_p$. This allows sensitive nodes to obtain better privacy protection, near-sensitive nodes to protect the sensitive node relationship network without completely losing effectiveness, and ordinary nodes to better serve downstream tasks. The MINE needs to calculate the mutual information between each pair of separately, which requires a lot of computing resources and time, when calculating high-dimensional data. We optimize the calculation of the MINE, using a single model to estimate the mutual information $\hat{I}_\theta(\hat{M}_n, \hat{M}_p)$ as a measure of the correlation between the sets before and after the privacy protection. By estimating mutual information, we can effectively judge whether the correlation degree between data before and after privacy protection exceeds the privacy threshold ε. Then, the framework adjusts the privacy protection degree of different nodes according to the mutual information, and re-evaluates the mutual information with the newly perturbed data, to obtain the trade-off between privacy and utility. Finally, when ε is satisfied, the protected embedding matrix $\hat{M}_p$ is obtained.

3. Preliminaries

In this section, we first review the graph embedding algorithm DeepWalk, which generates high-quality node representations in a compressed Euclidean space for various social network analysis tasks. Then, we introduce the mutual information neural

network estimator, which is employed to estimate mutual information when dealing with an unknown data distribution.

3.1. DeepWalk

A crucial element of DeepWalk involves utilizing the local neighbourhood structure of nodes to comprehend their relationships [9]. In the graph $G(V,E)$, where vertices V represent users and edges E denote relationship links between users. The generator selects a random node as the starting point of the random walk by performing uniform sampling while considering neighbouring nodes of the selected node as context. Subsequently, it learns the vector representation of the node by maximizing the co-occurrence probability between the context and the target node.

In detail, each node in G is embedded by the DeepWalk into a vector space, where each dimension corresponds to a feature of the node, which is denoted as $v_i \in \mathbb{R}^d$. The i-th row of M_n represents the d-dimensional embedding of the node v_i, which is written as [9]

$$M_n \in \mathbb{R}^{|V|\times d}, d \ll |V|. \tag{1}$$

Since grouping similar nodes in embedding can help learning algorithms achieve better performance in graph processing tasks, it is recommended to adopt the embedding algorithms like DeepWalk that have such grouping characteristics. Although node embedding is typically trained on a social network G, it encapsulates extensive structural information between nodes. An attacker may exploit this information to infer the network of relationships between sensitive nodes in G via membership inference attacks.

3.2. Mutual Information Neural Network Estimator(MINE)

Mutual information is a measure based on Shannon entropy that gauges the interdependence of two random variables. The mutual information between X and Y can be expressed as [29]

$$I(\mathrm{X};\mathrm{Y}) := H(\mathrm{X}) - H(\mathrm{X}|\mathrm{Y}), \tag{2}$$

where $H(\mathrm{X})$ represents the entropy of X, and $H(\mathrm{X}|\mathrm{Y})$ is the conditional entropy of X given Y.

When the data distribution is unknown, MINE selects a function that satisfies the integrability constraint: $T_\theta : \mathrm{X} \times \mathrm{Y} \to \mathbb{R}$, parameterized by a deep neural network, and estimates the mutual information between two random variables using the following formula [29]

$$\hat{I}_\theta(\mathrm{X};\mathrm{Y}) = \frac{1}{k}\sum_{i=1}^{k} T_\theta\left(x^{(i)}, y^{(i)}\right) - \log_2\left(\frac{1}{k}\sum_{i=1}^{k} e^{T_\theta\left(x^{(i)}, y^{(i)}\right)}\right), \tag{3}$$

where k represents the extraction of k mini-batch samples per batch from the sample.

It has been shown effective in previous research [28] for random datasets, but to estimate joint mutual information of multiple random variables, multiple models need to be trained. However, multiple T_θ will cause a large amount of data calculation, resulting in a giant and inefficient model.

4. Proposed Privacy and Utility Methods

In this section, we first provide a detailed explanation of how to label and group near-sensitive regions and ordinary regions by identifying sensitive nodes based on the characteristics of social networks after obtaining the embedding matrix. Next, we describe the modifications made to MINE to make it applicable to the graph embedding scenario, highlighting the key changes or adaptations. Finally, we introduce a privacy protection and utility trade-off algorithm that effectively safeguards the privacy of social network data while ensuring data utility. This algorithm eliminates the need for training multiple models, offering a more convenient and effective to privacy preservation. Finally, we discussed the time complexity of this framework.

4.1. Region Divisions Based on DeepWalk

After inputting the graph $G(V,E)$ adjacency matrix into the DeepWalk model, the network embedding matrix M_n is obtained, where row i can be denoted as v_i^n. Then, a set of all v_i^n can be represented as

$$\hat{M}_n = \{v_1^n, v_2^n, \ldots, v_i^n\}. \tag{4}$$

Algorithm 1 *RegionsDivide* ($\hat{M}_n$,$\hat{M}_{sensitive}$,α)

Input: embedding set $\hat{M}_n$, assigned sensitive node set $\hat{M}_{sensitive}$, number of near-sensitive node α.

Output: near-sensitive node set $\hat{M}_{near}$, ordinary node set $\hat{M}_{ordinary}$.

Initialize $\hat{M}_{near}, \hat{M}_{ordinary}$ as empty set
// Calculate the pairwise Euclidean distance between the sensitive node and all other nodes:
for $v_s \in \hat{M}_{sensitive}$
  for $v_j \in \hat{M}_n$ where v_j is a node
    Put $D(v_s, v_j) = \|v_s - v_j\|$ into D_{temp}
  end for
  $\hat{M}'_{near} \leftarrow$ top α of $Sort(D_{temp})$ in ascending order
end for
$\hat{M}'_{ordinary} \leftarrow \hat{M}_n \backslash \{\hat{M}_{sensitive}, \hat{M}'_{near}\}$
// Identify multi-privacy level nodes
for $v_i \in \hat{M}_n$
  if $v_i \in \hat{M}_{sensitive}$
    $\hat{M}_{near} \leftarrow \hat{M}'_{near} \backslash v_i$
    $\hat{M}_{ordinary} \leftarrow \hat{M}'_{ordinary} \backslash v_i$
  else if $v_i \in \hat{M}'_{near}$
    $\hat{M}_{ordinary} \leftarrow \hat{M}'_{ordinary} \backslash v_i$
  end if
end for
return $\hat{M}_{near}$,$\hat{M}_{ordinary}$

Given that the embedding model projects the graph into a low-dimensional space where the distance between nodes reflects their closeness in the original graph, we propose achieving privacy protection by perturbing distances rather than deleting nodes. However, the relationship networks of surrounding nodes to sensitive nodes could also reveal sensitive information. Therefore, determining the neighboring nodes around

sensitive nodes and perturbing them is equally important. This approach ensures that nodes with higher privacy requirements receive better privacy protection, while all nodes better serve the purpose of data analysis. The embedding distance is expressed as

$$D(v_i^n, v_j^n) = \|v_i^n - v_j^n\|, \tag{5}$$

where $v_i^n, v_j^n \in \hat{M}_n$, represented calculates the distance between the node and other nodes. Due to the uncertainty of the location of sensitive nodes in the social network, given a set of sensitive nodes $\hat{M}_{sensitive}$, we calculate the Euclidean distance between each sensitive node and other nodes, and the results are stored in D_{temp}. Given the number of near-sensitive node α, the α nodes closest to the sensitive nodes are selected as near-sensitive nodes through calculation. If α is too large, data utility will be lost. If α is too small, it will result in insufficient protection of sensitive nodes, so the size of α is crucial. We sort D_{temp} in ascending order by $Sort(D_{temp})$, label the top α nodes as closer to near-sensitive nodes, and store them into $\hat{M}'_{near}$. Then, we remove the given sensitive node set $\hat{M}_{sensitive}$ and the near-sensitive node set $\hat{M}'_{near}$ that meet the conditions from $\hat{M}_n$ to get $\hat{M}'_{ordinary}$

$$\hat{M}'_{ordinary} = \hat{M}_n \backslash \{\hat{M}_{sensitive}, \hat{M}'_{near}\}. \tag{6}$$

However, there may be situations where a node has three sensitivity levels simultaneously, for instance, if node 1 is a sensitive node itself and is also a near-sensitive node of node 2, and an ordinary node of node 3. In such cases, we consider the worst-case scenario, classifying it as a sensitive node based on its highest privacy level and removing it from other regions. Similarly, if a node is both near-sensitive and ordinary, we categorize it as a near-sensitive node. To address this situation, we divide the nodes with multiple privacy levels and remove the duplicate nodes to obtain the node set $\hat{M}_n$ after region division. The implementation is provided by Algorithm 1.

4.2. Regional Privacy Protection

Based on the sensitivity of each region, we apply different levels of privacy protection to achieve the goal of privacy protection. The Signal-to-Noise Ratio (*SNR*) [34] is

used to measure the relative intensity between signal and noise, defined as follows

$$SNR = \frac{P_S}{\sigma^2}, \tag{7}$$

where P_S is the signal power of the data, and σ^2 is the noise power. We define three privacy protection levels corresponding to different ranges of *SNR* values: sensitive region nodes are in the 1 to 3, near-sensitive region nodes are in the 4 to 6, and ordinary region nodes are in the 7 to 10.

For given *SNR*, we can calculate the P_S of the specific embedding matrix and obtain the σ_i^2 of Gaussian noise for each region for privacy protection, as follows

$$\hat{M}_p = \begin{Bmatrix} \hat{M}_{sensitive} + \mathcal{N}(0, \sigma_0^2 \mathrm{I}), \\ \hat{M}_{near} + \mathcal{N}(0, \sigma_1^2 \mathrm{I}), \\ \hat{M}_{ordinary} + \mathcal{N}(0, \sigma_2^2 \mathrm{I}) \end{Bmatrix}, \tag{8}$$

where $\mathcal{N}(0, \sigma_i^2 \mathrm{I})$ represents Gaussian noise with mean 0 and σ_i^2, and I is the identity matrix of size $d \times d$. It mean adding noise to each element in the three regions. The specific value of σ_i^2 is determined by *SNR* of the whole embedding matrix. After the *SNR* is determined, signal power P_S is calculated through the following formula

$$P_S = \frac{1}{V} \sum_{i=1}^{V} \sum_{d=1}^{d} \left| v_i^d \right|^2, \tag{9}$$

where V is the number of vectors in the embedding matrix, and v_i^d represents d-th element in embedding vector i. Then, we can set the accurate σ_i^2, accordingly based on subsequent Algorithm 3. The *SNR* for the specific region is determined based on privacy requirements ε in Section 4.4. Conducting distinct levels of privacy protection operations in each region yields the privacy-protected embedding matrix.

4.3. MINE for Embedding Vectors

Following the privacy protection procedure, it becomes essential to assess both data utility and privacy protection. Mutual information is a measure that gauges the interdependence of two random variables and serves as our means to quantify both privacy and utility.

The embedding matrix is a multi-dimensional feature matrix and can be viewed as multiple variables. However, calculating the mutual information of N variables requires evaluating the mutual information of each variable separately using (2), and then adding them up to obtain the final mutual information, as follows

$$I_{total}(\mathrm{X};\mathrm{Y}) := \sum_{n=1}^{N} \{H(\mathrm{X}_n) - H(\mathrm{X}_n|\mathrm{Y}_n)\}, \tag{10}$$

where (X;Y) represents N pairs of random variables, and X_n and Y_n represent the N-th pair of random variables. $H(\mathrm{X}_n)$ represents the entropy of X_n, and $H(\mathrm{X}_n|\mathrm{Y}_n)$ is the conditional entropy of X_n given Y_n. In other words, we first calculate the mutual information between each pair of random variables and then sum them to obtain the total mutual information.

Algorithm 2 MINE based on Graph Embedding: MINE-GE ($\hat{M}_n, \hat{M}_p, k, \eta$)

Input: original embedding $\hat{M}_n$, privacy embedding $\hat{M}_p$, batch size k, learning rate η .

Output: $\hat{I}_\theta(\hat{M}_n;\hat{M}_p)$

1: **repeat**

2: Sample k rows of vector $\left(m_n^{(1)},...,m_n^{(k)}\right)$ and vector $\left(m_p^{(1)},...,m_p^{(k)}\right)$ of the same nodes from $\hat{M}_n$ and $\hat{M}_p$ concatenate to l_n and l_p.

3: Compute mutual information using (12)

4: Calculate gradients with moving averages corrected for bias: $\hat{G}(\theta) \leftarrow \widetilde{\nabla}_\theta \hat{I}_\theta(\hat{M}_n;\hat{M}_p)$

5: Update system parameters: $\theta \leftarrow \theta + \eta \hat{G}(\theta)$

6: **until** convergence

Therefore, we can only calculate the mutual information of each variable individually through (3) and then accumulate them, as follows

$$\hat{I}_{total}(\mathrm{X};\mathrm{Y}) = \sum_{n=1}^{N} \hat{I}_\theta(\mathrm{X}_n;\mathrm{Y}_n). \tag{11}$$

This requires at least N neural network models to calculate and will generate N groups of different neural network parameters T_θ. This will cause a lot of waste of computing resources and time. In order to reduce the computational complexity, we

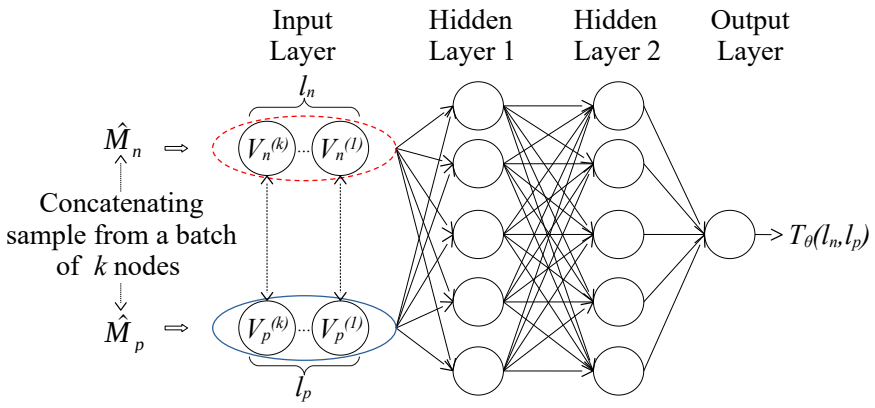


Figure 2: Neural network representation of our approximation function T_θ. The red dashed line ellipses represent unprocessed data and the blue solid line ellipses represent privacy-protected data. where l_n and l_p are a batch of data composed of corresponding the same k-row vector representations of k nodes in $\hat{M}_n$ and $\hat{M}_p$ respectively, as input to the model.

make MINE suitable for embedding matrices, which is called MINE-GE. By reasonable assumptions, we assume that each element of the embedding vector is i.i.d., because one dimension in the embedding space should not be correlated with another for a typical embedding algorithm. Otherwise, there is a waste in using extra dimensions. Therefore, we propose that each element be considered as a sample from the same distribution. We preprocess the embedding matrix by concatenating each embedded vector, forming a new embedding representation, and using this combination as the input for the neural network model. A direct benefit is that there will be one model to obtain the total mutual information of different random variables, speeding up the training. The mutual information estimation formula between $\hat{M}_n$ and $\hat{M}_p$ is

$$\hat{I}_\theta(\hat{M}_n;\hat{M}_p) = \frac{1}{k\times d}\sum_{i=1}^{k\times d} T_\theta\left(l_n^{(i)}, l_p^{(i)}\right) - \log_2\left(\frac{1}{k\times d}\sum_{i=1}^{k\times d} e^{T_\theta\left(l_n^{(i)}, l_p^{(i)}\right)}\right), \tag{12}$$

where k represents the batched k-row node vectorized representation, d represents the dimension of the vector, and we concatenate multiple vectors into two streams of samples l_n and l_p as the input for the neural network in Fig. 2. Here, we treat each dimension of the embedded vector as a sample. Subsequently, we input the samples one by one into MINE-GE and estimate their mutual information. The implementation is

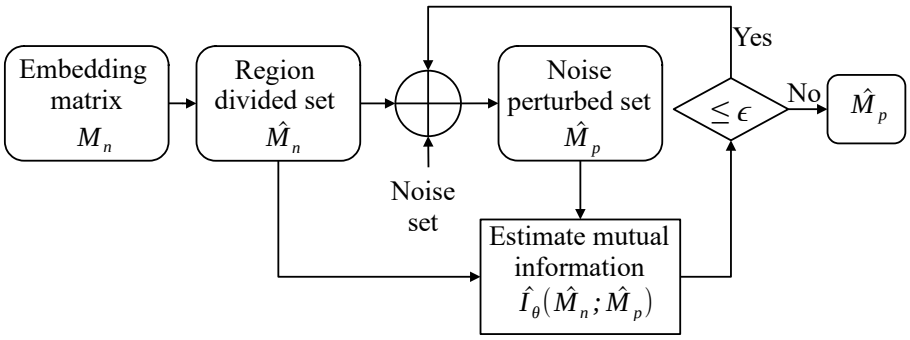


Figure 3: Proposed privacy and utility trade-off framework.

shown in Algorithm 2.

4.4. Privacy Utility Trade-off

Previous privacy protection methods retrain the embedding process by constantly perturbing the original graph, resulting in repeated computations to obtain a new embedding matrix. We proposed a privacy-utility trade-off framework that avoids disturbing the original graph and considers the privacy of all nodes, as shown in Fig. 3. This avoids repeated training of the embedding process and greatly reduces computational overhead.

Mutual information serves as an effective measure for quantifying both privacy protection and data utility. When estimating mutual information, it is essential to consider both the extent of privacy protection and the utility of the data. We adjust the privacy protection level through (12) estimated mutual information to ensure maximum data utility while maintaining privacy. Therefore, the optimization problem related to the trade-off between privacy and utility is defined as follows

$$\max_{\hat{I}_\theta(\hat{M}_n;\hat{M}_p)\leq\varepsilon} \hat{I}_\theta(\hat{M}_n;\hat{M}_p), \tag{13}$$

where $\hat{I}(\hat{M}_n;\hat{M}_p)$ is the estimated mutual information, and ε denotes the privacy threshold. This represents the calculation of mutual information between the privacy-preserving operation embedding matrices, maximizing mutual information while ensuring it is less than ε.

The privacy and utility trade-off, incorporating a privacy threshold, offers a solution to the optimization problem described in (13). We have proposed the trade-off between privacy and utility framework, which is essentially a process of re-sampling noise matrices in different regions based on mutual information. We estimate the mutual information $\hat{I}_\theta(\hat{M}_n;\hat{M}_p)$ using MINE-GE and adjust the noise level of each region based on mutual information to achieve a balance between privacy and utility.

The Algorithm 3 consists of three main parts: Firstly, the graph embedding algorithm is used to process the social network, and the low-dimension embedding matrix is obtained. Secondly, it identifies sensitive nodes based on the specific application scenario and labels and groups the graph into the sensitive region, near-sensitive region, and ordinary region based on Euclidean distance. Privacy protection operations are then applied to each region separately. Thirdly, it estimates the mutual information between $\hat{M}_n$ and $\hat{M}_p$ using MINE-GE and optimizes the privacy protection level using mutual information, achieving a trade-off between the privacy of the embedding matrix and the utility.

Our method enables the publication of graph data with privacy protection. After obtaining the published data, illegal data recipients cannot determine the sensitive, near-sensitive, and ordinary regions within the graph. Consequently, it becomes infeasible to extract a relationship network of sensitive nodes from the graph through simple denoising or inference attacks. Various privacy protection operations are carried out based on different regions, which can ensure the data utility of the graph-embedded data to a certain extent. Compared to conventional baseline techniques, our framework exhibits a distinct superiority in safeguarding the relationship network of sensitive nodes. The results of the multi-label classification task show that our framework effectively protects the privacy of the relationship network of sensitive nodes and the utility of the overall data.

Algorithm 3 DeepWalk assisted Regions Division and MINE-GE controlled Noise Addition

Input: graph $G(V,E)$, window length w, embedding dimension d, walks per vertex β, walk length t, sensitive node set $\hat{M}_{sensitive}$, number of near-sensitive node α, signal power P_S, Signal power of noise σ_i^2, SNR a_i, updating steps Δa_i, lower bound a_i^{LB} and upper bound a_i^{UB} for region i, near-sensitive region $\hat{M}_{near}$, ordinary region $\hat{M}_{ordinary}$, mini-batch size k, learning rate η, privacy threshold ε.

Output: $\hat{M}_p$

$\hat{M}_n \leftarrow DeepWalk(G,w,\beta,d,t)$

// Calculate the signal power of $\hat{M}_n$

$P_S = \frac{1}{V}\sum_{i=1}^{V}\sum_{d=1}^{d}\left|v_i^d\right|^2$ based on (9)

$\hat{M}_{near}, \hat{M}_{ordinary} \leftarrow RegionsDivide(\hat{M}_n, \hat{M}_{sensitive}, \alpha)$

// Calculate the noise power σ_i^2 in different regions, sample different Gaussian noise, and add them to three regions:

$\sigma_0^2 = P_S/a_0$, $\sigma_1^2 = P_S/a_1$, $\sigma_2^2 = P_S/a_2$

while $\hat{I_\theta}(\hat{M}_n;\hat{M}_p) \leq \varepsilon$ **do**

  Add noise of $\hat{M}_p$ based on (8)

  $\hat{I_\theta}(\hat{M}_n;\hat{M}_p) \leftarrow MINE\text{-}GE(\hat{M}_n,\hat{M}_p,k,\eta)$

  if $a_2^{LB} \leq a_2 \leq a_2^{UB}$ **then**

    $a_2 = a_2 + \Delta a_2$

    $\sigma_2^2 = P_S/a_2$

  else if $a_1^{LB} \leq a_1 \leq a_1^{UB}$ **then**

    $a_1 = a_1 + \Delta a_1$

    $\sigma_1^2 = P_S/a_1$

  else if $a_0^{LB} \leq a_0 \leq a_0^{UB}$ **then**

    $a_0 = a_0 + \Delta a_0$

    $\sigma_0^2 = P_S/a_0$

  end if

end while

return $\hat{M}_p$

5. Simulation Results

In this section, we present our simulation datasets, including BlogCatalog and Flickr, alongside other baseline methods. We then delve into the significance of the number of sensitive nodes within our methodology. To comprehensively understand the impact of different proportions of sensitive and near-sensitive regions on the overall partition, we conduct a simulation. Following this, we evaluate the effectiveness of privacy protection in our framework and other baseline methods by assessing the accuracy of membership inference attacks. In Algorithm 3, according to the privacy requirements of different regions, Gaussian noise with a mean value of 0 and different variance is added to each region for privacy protection. When we set $\varepsilon = 1$, we get the $SNR = 1$ of the sensitive region, the $SNR = 4$ of the near-sensitive region, and the SNR of ordinary nodes to be 10. Therefore, in order to better compare the privacy protection effect of each area, we set GN-1 to be the Gaussian noise when $SNR = 1$, GN-4 to be the Gaussian noise when $SNR = 4$, and GN-10 to be the Gaussian noise when $SNR = 10$. Subsequently, we adopted the same dataset and simulation procedure utilized by DeepWalk to perform a comparative analysis of our approach against other methods. Multi-label classification tasks serve as an effective means to validate the utility of data, with accuracy being a key metric. In certain scenarios, such as social media recommendation systems, user satisfaction with recommended content is directly influenced by the accuracy of the system in predicting node labels. Accurate label predictions enhance user experience and improve the practicality of the system. We conducted a comprehensive evaluation through a multi-label classification task and the AUC to validate the effectiveness of our method. We conducted a statistical analysis of the computational overhead of the experiment and the time of the algorithm. As shown in Table 2, we list all simulation parameters in our simulation.

5.1. Datasets, Baselines Methods and Attack Model

5.1.1. Datasets

In order to facilitate the comparison of our methods, we use the same parameters and datasets as DeepWalk.

Method	Parameters			
DeepWalk	$w = 10$	$d = 128$	$\beta = 80$	$t = 40$
MINE-GE	$\eta = 0.01$	$k = 4$	–	–
RegionsDivide	$\alpha = 50$	$\hat{M}_{sensitive} = \lceil V \times 1\% \rceil$	–	–
DPNE	$\varepsilon = 1$	d=128	–	–
Ours	$\varepsilon = 1$	a_0 =1	$a_1 = 4$	$a_2 = 10$
Ours_1	$\varepsilon = 1$	a_0 =2	$a_1 = 5$	$a_2 = 9$
Ours_2	$\varepsilon = 1$	a_0 =3	$a_1 = 6$	$a_2 = 7$

Table 2: Simulation parameters details

Name	$\lvert V \rvert$	$\lvert E \rvert$	$\lvert \mathscr{Y} \rvert$	Labels
BlogCatalog	10,312	333,983	39	Interests
Flickr	80,513	5,899,882	195	Groups

Table 3: Datasets details

- BlogCatalog is a social networks provided by bloggers. Tags represent the subject categories provided by the author.
- Flickr is a contact network between users of photo sharing websites. These tags represent user interest groups.

5.1.2. Baselines Methods

In order to facilitate the comparison of our methods, we use the same parameters, datasets, and multi-label classification experiments as DeepWalk.

- LPPGE [19] is a link-privacy preserved graph embedding method using adversarial learning, through related technologies, we reproduce LPPGE.
- DPNE [18] is a differentially private network embedding method based on DeepWalk as matrix factorization.
- GN-1 is $\mathscr{N}(0, P_S)$, that is Gaussian noise when $SNR = 1$. GN-4 is $\mathscr{N}(0, P_S/4)$, that is Gaussian noise when $SNR = 4$. GN-10 is $\mathscr{N}(0, P_S/10)$, that is Gaussian

noise when $SNR = 10$. Verify the impact of noise size on privacy protection and data utility.

5.1.3. Attack Model

In order to verify the effectiveness of privacy protection, we use the accuracy of membership inference attacks to evaluate the privacy protection effect.

- Membership Inference Attack is a privacy attack that seeks to determine whether a specific data point was included in the model's training set by analyzing the output of a machine learning model.

5.2. Performance of Region Divisions

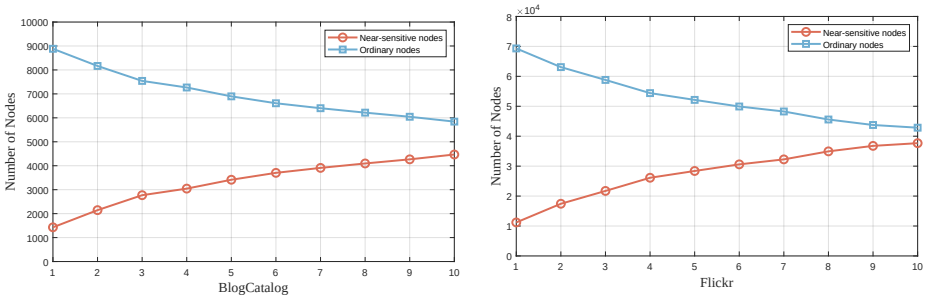


Figure 4: When the number of near-sensitive nodes ranges from 20 to 200 with an interval of 20 the number of nodes in the sensitive and ordinary region when the number of sensitive nodes is 1% in BLogCatalog and Flickr.

In our approach, the proportion of sensitive nodes significantly influences overall data utility. While a higher proportion of sensitive and near-sensitive nodes improves privacy protection, it may compromise data utility. Therefore, determining the appropriate proportion of sensitive nodes is crucial. To investigate the impact of the ratio of near-sensitive nodes on the overall data, we set the ratio of sensitive regions to 1% of the total nodes and systematically expanded the scope of near-sensitive nodes. The results, depicted in Fig. 4, reveal that as the number of near-sensitive nodes increases, the corresponding nodes in the near-sensitive region also increase, while those in the ordinary region decrease. As a result, in the *RegionDivide* algorithm, we fix the sensitive region at 1%, and the number of near-sensitive nodes, denoted as α, is set to 50.

In practical scenarios, adjusting the proportion of the near-sensitive region according to specific circumstances allows *RegionDivide* to optimize its effectiveness.

5.3. Performance of Privacy Protection

As shown in Fig. 5, Gaussian noise $\mathcal{N}(0, P_S/SNR)$, where $SNR = 1$ to 10 is added to the data processed by DeepWalk, and the mutual information value between this perturbed data and the original graph-embedded data is calculated using MINE-GE. According to the formula's definition, when the P_S value is fixed, a larger SNR corresponds to a smaller σ^2 value. Therefore, The smaller the added Gaussian noise, the lower the degree of privacy protection. The greater the added Gaussian noise, the higher the degree of privacy protection, but the data utility will not be guaranteed. Simulations reveal that when the mutual information between processed data and graph-embedded data exceeds 1, data utility can be assured, yet it might lead to privacy leakage. We further compare the privacy protection effects of our method with other baseline methods in membership inference attacks.

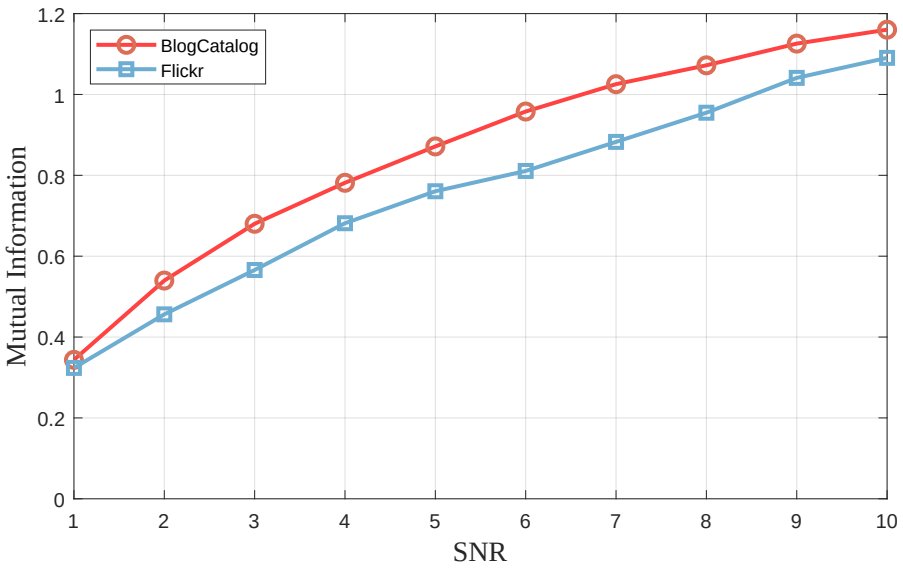


Figure 5: Mutual information between DeepWalk and the results of the privacy protection method of adding Gaussian noise $\mathcal{N}(0, P_S/M)$, where $M = 1$ to 10, that is Gaussian noise when $SNR = 1$ to 10.

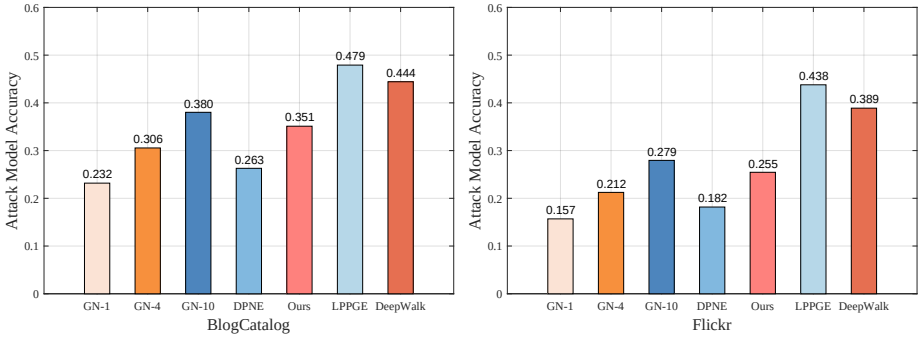


Figure 6: The accuracy of membership inference attack on BlogCatalog and Flickr.

Through membership inference attacks on privacy-protected data, the results are shown in Fig. 6. It can be observed that the accuracy of the attack model against GN-1, GN-4, and DPNE is significantly lower, indicating a good privacy protection effect. However, under privacy protection with high noise levels, data utility cannot be guaranteed. GN-10 ensures the utility of data, but the privacy of data is not guaranteed. LPPGE is even better than DeepWalk in accuracy, which verifies that the method of reconstructing the graph for privacy protection retains the relevant information of the original graph, which leaks the privacy of the original graph to a certain extent, and the availability of the data is not guaranteed. Our method provides high-level privacy protection for the relational network of sensitive nodes, reduces the accuracy of the attack model, and proves its success in privacy protection while ensuring data utility. The verification of data utility is described in detail in the subsequent multi-label classification task.

From Fig. 6, it can be observed that the accuracy of our method in the membership inference attack model is only slightly higher than GN-4 and DPNE. However, due to Algorithm 3 performing different levels of privacy protection operations in different regions, it has a privacy protection advantage in the relational network concerning sensitive nodes. This is because it introduces larger Gaussian noise in the near-sensitive region and smaller Gaussian noise in the ordinary region. This targeted privacy protection can make the need to protect the relationship between the network to get better privacy protection.

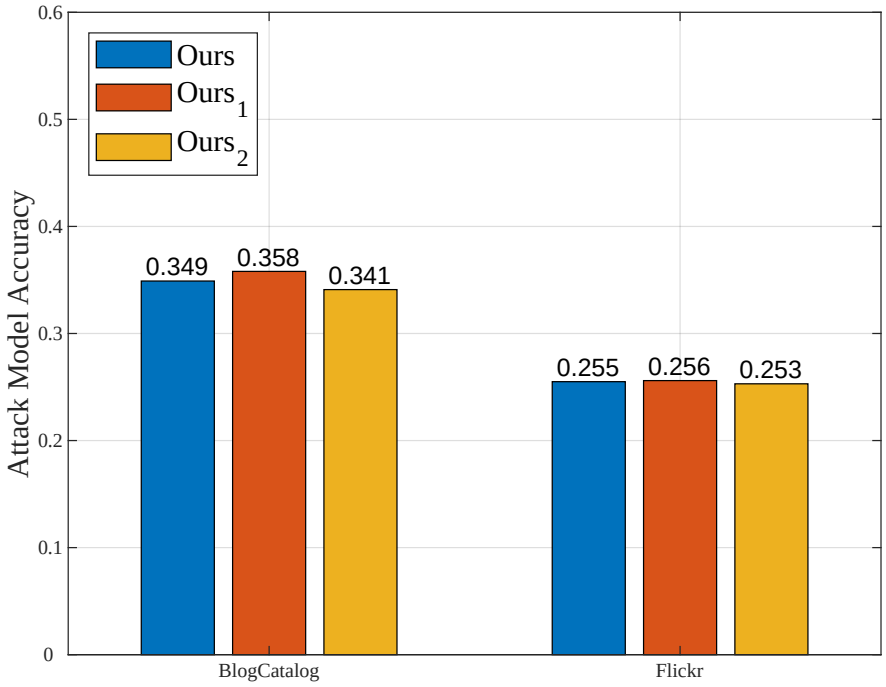


Figure 7: The accuracy of membership inference attack.

We found that when we set $\varepsilon = 1$, different privacy protection scenarios can be applied to three regions. For example, in the first scenario, we get $SNR = 1$ of the sensitive region, the $SNR = 4$ of the near-sensitive region, and the $SNR = 10$ of the ordinary region, recorded as Ours. In the second scenario, we get $SNR = 2$ of the sensitive region, the $SNR = 5$ of the near-sensitive region, and the $SNR = 9$ of the ordinary region, recorded as Ours_1. In the third scenario, we get $SNR = 3$ of the sensitive region, the $SNR = 6$ of the near-sensitive region, and the $SNR = 7$ of the ordinary region, recorded as Ours_2. As shown in Fig. 7, when we set the privacy threshold $\varepsilon = 1$, the membership inference attack accuracy of the three situations are (0.349, 0.358, 0.341) in the BlogCatalog datasets and (0.255, 0.256, 0.253) in the Flickr datasets. The element difference of each tuple does not exceed 0.02, this shows that three different privacy protection levels can achieve privacy protection effects.

5.4. Performance of Privacy and Utility Trade-off

In the multi-label classification tasks, we randomly select a portion (T_R) of labeled nodes as the training datasets, while the remaining nodes serve as the testing dataset. We perform this process ten times and present the preliminary results directly, showing the average performance of both Macro-F1 and Micro-F1 scores.

Table 4: Multi-label classification results in BlogCatalog.

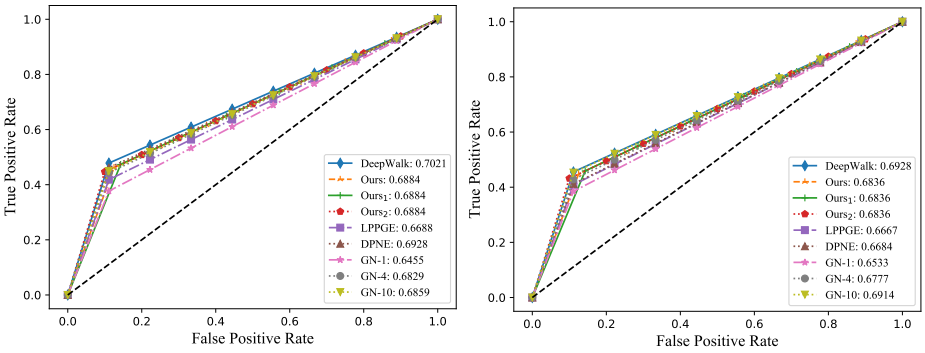
	Method	10 %	20 %	30 %	40 %	50 %	60 %	70 %	80 %	90 %
Micro-F1(%)	DeepWalk	36.09	38.87	40.30	41.16	41.68	42.11	42.32	42.64	42.97
	Ours	33.68	36.42	37.99	38.69	39.28	39.58	39.81	40.26	40.84
	$Ours_1$	34.15	36.66	38.00	39.03	39.64	40.16	40.52	40.77	40.95
	$Ours_2$	33.48	36.19	37.47	38.54	39.22	39.66	39.96	40.28	40.89
	LPPGE	31.44	34.27	35.82	36.50	37.17	37.56	37.80	37.88	38.40
	DPNE	27.96	30.95	32.77	33.73	34.53	35.10	35.49	35.84	36.67
	GN-10	34.63	36.87	38.26	39.01	39.61	39.90	40.35	40.46	40.83
	GN-4	31.46	34.29	35.87	36.36	36.91	37.48	37.99	38.47	38.79
	GN-1	25.52	28.82	30.40	31.61	32.26	32.86	33.27	34.09	34.25
Macro-F1(%)	DeepWalk	21.04	24.27	25.94	26.92	27.67	28.23	28.49	28.55	28.56
	Ours	19.48	22.48	23.98	24.94	25.64	26.05	26.27	26.36	26.70
	$Ours_1$	19.54	22.23	23.90	25.04	26.05	26.68	27.03	27.34	27.63
	$Ours_2$	19.01	22.07	23.53	24.67	25.30	25.91	26.47	27.08	27.82
	LPPGE	16.81	19.77	21.20	22.30	22.88	23.28	23.69	23.52	23.74
	DPNE	15.33	17.90	19.20	20.09	20.65	21.18	21.36	21.74	22.21
	GN-10	20.08	22.59	24.22	24.96	25.56	25.89	26.18	26.36	26.83
	GN-4	17.53	20.56	22.09	22.73	23.29	23.64	23.95	24.73	25.06
	GN-1	13.85	16.47	17.61	18.57	19.01	19.77	20.26	20.62	20.71

Table 4 and Table 5 lists the simulation results of multi-label classification after privacy protection on BlogCatalog and Flickr datasets. In the GN-1 privacy-preserving method, the recognition rate in the multi-label classification task is significantly reduced by nearly 11% compared with the DeepWalk, which the privacy of the embedding matrix is protected, the loss of data utility is significant. In the GN-10 privacy-preserving method, the recognition rate in the multi-label classification task is only reduced by about 2% compared with DeepWalk, which could potentially lead to privacy breaches despite improved data utility. In the multi-label classification task, DPNE is reduced by almost 8% on Micro-F1 and 6% on Macro-F1 compared to the DeepWalk embedding matrix, which is because DPNE prioritizes privacy protection, resulting in

Table 5: Multi-label classification results in Flickr.

	Method	1 %	2 %	3 %	4 %	5 %	6 %	7 %	8 %	9 %
Micro-F1(%)	DeepWalk	32.23	34.63	35.96	36.83	37.40	37.84	38.25	38.58	38.86
	Ours	30.19	32.89	34.21	35.02	35.58	36.08	36.44	36.77	37.05
	$Ours_1$	30.51	32.82	34.12	35.02	35.67	36.17	36.64	36.93	37.16
	$Ours_2$	30.26	32.72	34.16	35.04	35.69	36.21	36.54	36.92	37.19
	LPPGE	27.56	29.83	31.21	32.11	32.71	33.24	33.60	33.90	34.18
	DPNE	26.53	28.94	30.38	31.28	31.90	32.46	32.92	33.32	33.62
	GN-10	31.36	34.18	35.57	36.48	36.97	37.48	37.97	38.25	38.59
	GN-4	29.09	31.67	33.03	33.94	34.62	35.10	35.50	35.85	36.29
	GN-1	23.46	25.80	27.41	28.42	29.20	29.80	30.29	30.72	31.07
Macro-F1(%)	DeepWalk	13.06	16.72	19.13	20.84	21.93	22.78	23.62	24.28	24.86
	Ours	11.99	15.75	17.91	19.56	20.63	21.58	22.31	22.90	23.43
	$Ours_1$	12.28	15.75	17.98	19.62	20.71	21.57	22.42	23.04	23.42
	$Ours_2$	11.91	15.61	17.92	19.50	20.80	21.76	22.33	23.03	23.56
	LPPGE	9.39	11.97	13.88	15.13	16.22	17.08	17.75	18.23	18.75
	DPNE	9.74	12.94	14.92	16.27	17.33	18.17	18.94	19.54	19.98
	GN-10	12.86	16.49	18.81	20.44	21.54	22.36	23.19	24.03	24.27
	GN-4	11.49	14.83	17.01	18.64	19.73	20.64	21.31	21.89	22.72
	GN-1	8.29	10.62	12.59	13.87	14.95	15.80	16.48	17.09	17.55

some loss of data utility. LPPGE is nearly 6% lower than the DeepWalk embedding matrix on Micro-F1 and approximately 4% lower on Macro-F1. In the multi-label classification task, our method is only approximately 2% lower than DeepWalk on Micro-F1 and Macro-F1. And as we can see, the results of the multi-label classification task for the three privacy protection schemes Ours, $Ours_1$ and $Ours_2$ added through our framework are essentially consistent. This also validates that our framework can adjust the degree of privacy protection based on the actual situation of nodes and the privacy requirements of user nodes, thereby meeting the practical needs of various scenarios. Subsequently, we plotted the AUC for the multi-label classification task. As shown in Fig. 8, the AUC of the three types of privacy protection through our framework is 0.6884, which is higher than other baseline methods and only slightly lower than DeepWalk and LPPGE that do not perform privacy protection operations. However, the LPPGE method performs much worse than our method in membership inference attacks. It can be seen that when the greater the noise is added, the AUC is lower, and GN-1 is only 0.6455, which shows that its privacy protection effect is very good. It



(a) BlogCatalog (b) Flickr

Figure 8: AUC for multi-label classification tasks.

also indirectly shows that our privacy protection method performs well in protecting sensitive nodes. This shows that our privacy-preserving framework protects privacy while ensuring data utility.

The simulation results of multi-label classification tasks demonstrate that our method outperforms other methods in terms of data utility while only slightly lagging behind GN-10. However, it's important to note that GN-10 sacrifices a significant amount of privacy protection for the sake of data utility. Given the unique label and group approach used in our method, it incorporates a higher level of privacy protection in near-sensitive regions. Therefore, compared to other privacy protection methods, our method excels in safeguarding the relevant information and actual social relationships of sensitive nodes, providing advantages in privacy protection while ensuring the overall utility of the dataset.

Although the framework achieves good performance in the multi-label classification tasks, still it has some limitations. Firstly, the fixed approach may not be suitable for cases where groups have different sensitivities. Secondly, Gaussian noise was used in the perturbation process, which may not be appropriate for all the cases. Thirdly, there is a performance gap between our trade-off framework and the original DeepWalk that does not involve protection. It would be desirable to achieve an embedding that has the classification performance as closest to DeepWalk while still offering protection.

5.5. Time Cost

To compare the time spent between our framework and other baseline methods, we conducted all simulation methods on the same platform in Table 6. For datasets BlogCatalog and Flickr, each algorithm was executed ten times on each dataset, or the middle results. The average in Table 7 reveals that the graph embedding process in DeepWalk is notably time-consuming. It has an increasing time as the graph size increases. Embedding the Flickr takes up to seven times longer than the BlogCatalog. This also illustrates that the traditional privacy protection method of deleting the original nodes in the graph and then retraining the embedding model is very time-consuming. For instance, retraining these two datasets merely twice would demand nearly doubled time if the graph is very large.

Name	Version
Operating System	Windows 11
CPU	Intel(R) Core(TM) i7-12700F
GPU	NVIDIA RTX 3070 8GB
RAM	16GB
Python	3.6.2

Table 6: Platform details

Datasets	*DeepWalk*	*RegionsDivide*	MINE-GE	Total
BlogCatalog	415.05s	3.75s	38.53s	495.84s
Flickr	2868.24s	204.81s	358.45s	3789.83s

Table 7: Time cost

Meanwhile, the *RegionsDivide* algorithm only needs to run once after obtaining the embedding matrix through the embedding algorithm. Throughout the privacy-utility trade-off process, we repetitively compute the mutual information between the privacy-protected data and the original embedding data. For each MINE-GE training, it takes 38.53s and 358.45s for the two datasets, respectively. Although the computation time

scales with the dataset size, it remains significantly smaller than the time cost of re-training the embedding model. Notably, for $\varepsilon = 1$, the total time required to execute the entire framework is $495.84s$ and $3789.83s$, respectively. This includes one-time DeepWalk training and *RegionsDivide* as well as multiple rounds of MINE-GE.

Compared with the previous privacy protection method of deleting nodes and re-training of embedding, it takes a corresponding amount of time to execute the embedding algorithm once, so k times of re-training in the privacy utility trade-off process requires $k \times 415.05s$ and $k \times 2868.24s$ respectively. For any $k \geq 2$, $k \times 415.05s \gg 495.84s$, and $k \times 2868.24s \gg 3789.83s$, the time cost of our framework investment is considerably less than the time cost associated with retraining the embedding model after removing original nodes.

6. Conclusion and Future Work

In this paper, we present an innovative approach to tackle the privacy protection challenges associated with graph embedding. This method targets all nodes for privacy protection while balancing utility without retraining embeddings, dividing the embedding data into three different parts: sensitive, near-sensitive, and ordinary regions. This partitioning method allows us to provide higher privacy protection for sensitive nodes while retaining higher data utility for ordinary nodes. Leveraging this concept, We propose a privacy protection and utility trade-off framework for social networks based on DeepWalk-assisted regionalization and MINE-GE controlled noise addition. Near-sensitive and ordinary regions are determined by defining sensitive nodes and calculating the Euclidean distance between sensitive nodes and other nodes. Different noises are added to sensitive, near-sensitive, and ordinary regions to protect privacy. Using mutual information to quantify privacy and utility, we employ MINE-GE to calculate the mutual information between these regions to optimize our method's privacy protection effect. This ensures the privacy of sensitive and near-sensitive regions in graph embedding and the overall utility of embedded data. Finally, through membership inference attacks comparing the privacy protection effectiveness of our method with other baseline methods, we verify that our method has superiority in protecting the

relational network concerning sensitive nodes. By comparing the multi-label classification results of our method, DPNE, LPPGE, and other methods through multi-label classification tasks, we demonstrate that our method can achieve a trade-off between the privacy and utility of data.

To achieve a protected embedding performance near DeepWalk in future work, we can try to identify near-sensitive nodes by varying distances to dynamically incorporate group differences in protection strength, and privacy-preserving operations may be improved by performing more time-consuming optimizations over continuous *SNR* rather than discrete ones. Furthermore, different noise distributions as well as their mixtures may be used by the noise perturbations in future work to achieve better protection while having the same utility performance. Meanwhile, when the embedding variable is correlated in some embedding algorithms, future work may need to resort to multi-model-based MINE for more detailed and reliable information on the node relationship. But still, our single model can capture the typicality of the relationships of the nodes.

Acknowledgement

This work was supported by Guizhou University through the project of Secure Encryption Mechanisms of Spatially Embedded Networks under Guizhou University Natural Science Special Grant No. (2021) 30.

References

[1] J. S. He, M. Han, S. Ji, T. Du, Z. Li, Spreading social influence with both positive and negative opinions in online networks, Big Data Mining and Analytics 2 (2) (2019) 100–117. doi:10.26599/BDMA.2018.9020034.

[2] X. Zheng, G. Luo, Z. Cai, A fair mechanism for private data publication in online social networks, IEEE Transactions on Network Science and Engineering 7 (2) (2020) 880–891. doi:10.1109/TNSE.2018.2801798.

[3] X. Zheng, Z. Cai, G. Luo, L. Tian, X. Bai, Privacy-preserved community discovery in online social networks, Future Generation Computer Systems 93 (2019) 1002–1009. doi:https://doi.org/10.1016/j.future.2018.04.020.

[4] M. Adams, Big data and individual privacy in the age of the internet of things, Technology Innovation Management Review 7 (2017) 12–24. doi:http://doi.org/10.22215/timreview/1067.

[5] X. Han, Y. Yang, L. Wang, J. Wu, Privacy-preserving network embedding against private link inference attacks, IEEE Transactions on Dependable and Secure Computing (2023) 1–13doi:10.1109/TDSC.2023.3264110.

[6] M. Siddula, Y. Li, X. Cheng, Z. Tian, Z. Cai, Anonymization in online social networks based on enhanced equi-cardinal clustering, IEEE Transactions on Computational Social Systems 6 (4) (2019) 809–820. doi:10.1109/TCSS.2019.2928324.

[7] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, Vol. 14, 2001.

[8] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326. doi:10.1126/science.290.5500.2323.

[9] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proc. of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, 2014, p. 701–710. doi:10.1145/2623330.2623732.

[10] T. Mikolov, K. Chen, G. S. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: International Conference on Learning Representations, Scottsdale, AZ, USA, 2013.

[11] A. Sala, X. Zhao, C. Wilson, H. Zheng, B. Y. Zhao, Sharing graphs using differentially private graph models, in: Proc. of the 2011 ACM SIGCOMM Confer-

ence on Internet Measurement Conference, Association for Computing Machinery, Berlin, Germany, 2011, p. 81–98. doi:10.1145/2068816.2068825.

[12] Y. Wang, X. Wu, Preserving differential privacy in degree-correlation based graph generation, Transactions on Data Privacy 6 (2) (2013) 127–145.

[13] J. Hua, C. Xia, S. Zhong, Differentially private matrix factorization, in: Proc. of the 24th International Conference on Artificial Intelligence, Buenos Aires, Argentina, 2015, p. 1763–1770.

[14] Z. Liu, Y.-X. Wang, A. Smola, Fast differentially private matrix factorization, in: Proc. of the 9th ACM Conference on Recommender Systems, Vienna Austria, 2015, p. 171–178. doi:10.1145/2792838.2800191.

[15] F. Zhang, V. E. Lee, K.-K. Raymond Choo, Jo-dpmf: Differentially private matrix factorization learning through joint optimization, Information Sciences 467 (2018) 271–281. doi:https://doi.org/10.1016/j.ins.2018.07.070.

[16] P. Jain, O. D. Thakkar, A. Thakurta, Differentially private matrix completion revisited, in: Proc. 35th International Conference on Machine Learning (ICML), PMLR, Stockholm, SWEDEN, 2018, pp. 2215–2224.

[17] M. Jaggi, M. Sulovsk, et al., A simple algorithm for nuclear norm regularized problems, in: Proc. of the 27th international conference on machine learning (ICML-10), Haifa Israel, 2010, pp. 471–478.

[18] D. Xu, S. Yuan, X. Wu, H. Phan, DPNE: Differentially private network embedding, in: Proc. Advances in Knowledge Discovery and Data Mining:Advances in Knowledge Discovery and Data Mining: 22nd Pacific-Asia Conference, Springer, Deakin Univ, Melbourne, Australia, 2018, pp. 235–246.

[19] K. Zhang, Z. Tian, Z. Cai, D. Seo, Link-privacy preserving graph embedding data publication with adversarial learning, Tsinghua Science and Technology 27 (2) (2022) 244–256. doi:10.26599/TST.2021.9010015.

[20] M. A. El-Sayed, T. A.-E. Hafeez, New edge detection technique based on the shannon entropy in gray level images, arXiv preprint arXiv:1211.2502 (12 November 2012).

[21] A. Badawy, J. A. Fisteus, T. M. Mahmoud, T. Abd El-Hafeez, Topic extraction and interactive knowledge graphs for learning resources, Sustainability 14 (1) (2021) 226.

[22] I.-C. Hsieh, C.-T. Li, Netfense: Adversarial defenses against privacy attacks on neural networks for graph data, IEEE Transactions on Knowledge and Data Engineering 35 (1) (2023) 796–809. doi:10.1109/TKDE.2021.3087515.

[23] X. He, R. Wen, Y. Wu, M. Backes, Y. Shen, Y. Zhang, Node-level membership inference attacks against graph neural networks, arXiv preprint arXiv:2102.05429 (10 February 2021).

[24] P. Liao, H. Zhao, K. Xu, T. Jaakkola, G. J. Gordon, S. Jegelka, R. Salakhutdinov, Information obfuscation of graph neural networks, in: Proc. International Conference on Machine Learning, PMLR, Virtual, 2021, pp. 6600–6610.

[25] M. Ellers, M. Cochez, T. Schumacher, M. Strohmaier, F. Lemmerich, Privacy attacks on network embeddings, ArXiv abs/1912.10979 (23 December 2019).

[26] J. McAuley, J. Leskovec, Hidden factors and hidden topics: understanding rating dimensions with review text, in: Proc. of the 7th ACM conference on Recommender systems, Hong Kong, China, 2013, pp. 165–172.

[27] M. De Choudhury, S. Counts, E. Horvitz, Social media as a measurement tool of depression in populations, in: Proc. of the 5th annual ACM web science conference, New York, NY, USA, 2013, pp. 47–56.

[28] Q. Wu, J. Tang, S. Dang, G. Chen, Data privacy and utility trade-off based on mutual information neural estimator, Expert Systems with Applications 207 (2022) 118012. doi:https://doi.org/10.1016/j.eswa.2022.118012.

[29] M. I. Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, R. D. Hjelm, A. C. Courville, MINE: Mutual information neural estimation, in: 35th International Conference on Machine Learning (ICML), Vol. 80, PMLR, Stockholmsmässan, Sweden, 2018, pp. 531–540.

[30] A. Abdelmged, A. Tarek, A.-H. Seddik, M. Shaimaa, Improving ZOH image steganography method by using braille method, International Journal of Computer Applications 975 (2016) 8887.

[31] A. AA, T. AA, A.-H. Seddik, M. Shaimaa, New image steganography method using zero order hold zooming, International Journal of Computer Applications 975 (2016) 8887.

[32] T. Radhika, A. Chandrasekar, V. Vijayakumar, Q. Zhu, Analysis of markovian jump stochastic Cohen–Grossberg BAM neural networks with time delays for exponential input-to-state stability, Neural Processing Letters (2023) 1–18.

[33] Y. Cao, A. Chandrasekar, T. Radhika, V. Vijayakumar, Input-to-state stability of stochastic markovian jump genetic regulatory networks, Mathematics and Computers in Simulation (August 2023).

[34] R. Fritschek, R. F. Schaefer, G. Wunder, Deep learning for channel coding via neural mutual information estimation, in: Proc. 2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Cannes, France, 2019, pp. 1–5. doi:10.1109/SPAWC.2019.8815464.