Local Differential Privacy in Graph Neural Networks: a Reconstruction Approach

Karuna Bhaila
University of Arkansas
kbhaila@uark.edu

Wen Huang
University of Arknasas
wenhuang@uark.edu

Yongkai Wu
Clemson University
yongkaw@clemson.edu

Xintao Wu
University of Arkansas
xintaowu@uark.edu

Abstract

Graph Neural Networks have achieved tremendous success in modeling complex graph data in a variety of applications. However, there are limited studies investigating privacy protection in GNNs. In this work, we propose a learning framework that can provide node-level privacy, while incurring low utility loss. We focus on a decentralized notion of Differential Privacy, namely Local Differential Privacy, and apply randomization mechanisms to perturb both feature and label data before being collected by a central server for model training. Specifically, we investigate the application of randomization mechanisms in high-dimensional feature settings and propose an LDP protocol with strict privacy guarantees. Based on frequency estimation in statistical analysis of randomized data, we develop reconstruction methods to approximate features and labels from perturbed data. We also formulate this learning framework to utilize frequency estimates of graph clusters to supervise the training procedure at a sub-graph level. Extensive experiments on real-world and semi-synthetic datasets demonstrate the validity of our proposed model.

1 Introduction

Graph data are ubiquitous in the modern world allowing graph-structured representation for complex data in the realm of social networks, finance, biology, and so on. Graph Neural Networks (GNNs) have been widely adopted to model the expressive nature of such graph-structured data [37]. GNNs rely on *message-passing* mechanisms to propagate information between graph nodes and output embeddings that encode both node and neighborhood features aggregated using graph adjacency information. These embeddings are used in predictive downstream tasks for meaningful applications such as drug discovery, traffic prediction, recommendation, and so on. This widespread prevalence of GNNs, however, raises concerns regarding the privacy of sensitive information whose leakage may lead to undesirable and even harmful consequences. GNNs have been shown to be vulnerable to several privacy risks including membership inference [24], link re-identification [16], and attribute disclosure [41]. The risks are considerably higher in GNNs compared to traditional learning models due to the presence of additional graph structure information [24]. To ensure compliance with legal data protection guidelines [22] and for the protection of user privacy, GNNs must thus be trained and deployed in a privacy-preserving manner.

In this paper, we aim to address such privacy concerns in GNNs. We focus on a specific scenario of node privacy wherein node-level features and labels are held locally by each user and the global graph structure is available to a central server. The server could benefit from users' feature data which paired with graph topology can be utilized for embedding generation and/or predictive modeling via

NeurIPS 2023 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2023).

GNNs. However, collecting user feature and label data, possibly containing sensitive and identifying information, may incur serious privacy issues. To this end, Local Differential Privacy (LDP) [18] is often adopted during data collection for model training or releasing statistics privately [8]. Furthermore, it has been deployed in large-scale data-gathering of user behavior and usage statistics at Apple [1] and Google [13] motivating the integration of LDP in data collection for GNNs as well.

Challenges The main challenge in training GNNs with privately collected data is the utility-privacy trade-off of differentially private mechanisms [33]. As a whole, data randomized at an individual level oftentimes misrepresents the population-level distribution. A learning model that learns feature and label correlation from this data may overfit the noise and achieve sub-par performance on predictive and analytical tasks with unseen data [38]. Furthermore, since GNNs propagate information throughout the graph to output node embeddings, the quality of the embeddings also suffers due to additive noise present in the training data after applying LDP mechanisms.

Prior Work A few recent works have attempted to address node privacy in GNNs [9, 23] by enforcing privacy during training and/or model release. This potentially puts user information at risk. Sajadmanesh et al. [29] propose a node-level LDP framework in the distributed setting where features and labels are held private by the user and the graph structure is known to the server. They propose an LDP protocol called multi-bit mechanism to perturb node features by extending the 1-bit mechanism [10] to multidimensional features. The multi-bit mechanism randomly selects a subset of features for each user, transforms each selected feature value to either 1 or -1, and indiscriminately reports the value 0 for the remaining ones. To protect label privacy, node labels are perturbed using Randomized Response (RR) [35]. A GCN-based multi-hop aggregator is then prepended to the GNN model for implicit denoising of both features and labels. They further implement forward loss correction [26] to supervise the learning process in the presence of noisy labels. However, the multi-bit mechanism potentially results in a huge loss of information, especially considering that the size of the sampled feature subset is set to 1 as per the analysis presented in the paper. The model is evaluated on several graph datasets with high-dimensional binary features where each feature has around 99% zero values (shown in Table 2 in appendix). This inadvertently aids to reduce variance during aggregation of features perturbed via the multi-bit mechanism. This may not be the case during deployment in real world which could significantly affect the privacy-utility trade-off.

Contributions In this work, we propose RGNN, a novel reconstruction-based learning framework that ensures LDP for nodes while incurring low utility loss. To protect feature privacy, we extend previous work by Arcolezi et al. [4] and implement Generalized Randomized Response with Feature Sampling (GRR-FS), a randomization framework with provable LDP guarantees. To minimize utility loss caused by randomization, we propose reconstruction methods that approximate true features and label distributions from the perturbed data via theoretically derived frequency estimation techniques. We leverage graph homophily and use these methods to estimate data distributions at a sub-graph level and ultimately at the node level. We further introduce propagation during reconstruction to reduce estimation variance. We also formulate this learning framework to include frequency estimates of graph clusters to supervise the training procedure. The proposed framework can be paired with any GNN architecture and does not require private data for training or validation. We perform extensive experiments on real-world and semi-synthetic datasets for the task of transductive node classification under varying privacy budgets. Empirical results show our method's effectiveness over baselines.

2 Related Work

Privacy leakage in GNNs has become an unavoidable concern due to real-world implications of models trained on potentially sensitive data. In order to protect against privacy leakage and attacks, various attempts have been made to develop privacy-preserving GNN algorithms, including the extension of Differential Privacy (DP) to GNNs. [36] proposes an edge-level DP algorithm by adding noise to the adjacency matrix as a pre-processing step. However, the edge-DP problem is different from the node privacy setting explored in this paper where the graph topology is non-private but the node features and labels are locally private. [40] proposes a node-level DP algorithm that decouples message-passing from feature aggregation and uses an approximate personalized PageRank to perform feature transformation instead of message-passing. This algorithm requires a trusted aggregator to compute the approximate personalized PageRank matrix from private data and is more suited for private release of trained GNN models and their outputs than ensuring user-level privacy. [9] proposes a node-level DP algorithm by integrating sub-graph sampling and the standard

DP-SGD [2] algorithm into the training framework to update GNN parameters, but this approach also requires a trusted aggregator to perform sampling before model training and is limited to a 1-layer GNN model. [23] utilizes the student-teacher training workflow from the PATE framework [25] to release a student GNN trained on public data and partly private labels obtained via the teacher GNN with DP guarantees. However, their framework necessitates the availability of public graph data which is not possible under LDP constraints considered in this work. [31] proposes a randomization mechanism to ensure node-level and edge-level DP by optimizing the magnitude of noise injected into nodes' features and the graph structure. However, node labels are considered non-private in their setting. Our work is most closely related to that of [29] where node features and labels are individually perturbed before sending to a server for training. They utilize GCN-based multi-hop aggregation for denoising features, and labels and incorporate forward loss correction [26] to facilitate training with noisy labels. However, implicit denoising via aggregation may not be adequate to obtain accurate graph signals due to propagation of noise during aggregation.

3 Preliminaries

We use $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to denote an undirected graph where $\mathcal{V}$ is a set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ defines graph edges represented by an adjacency matrix $\mathbf{A} \in \{0,1\}^{N \times N}$ s.t. $\mathbf{A}_{u,v} = 1$ if an edge exists between nodes u and v. $\mathbf{X} = \{\mathbf{x}_1, \dots \mathbf{x}_N\}$ represents node features, where $\mathbf{x}_v \in \mathbb{R}^d$ is the d-dimensional feature vector of node v and we use $\mathbf{X}_i \in \mathbb{R}^N$ to denote the i-th feature column. In a transductive setting, a fraction of nodes denoted $\mathcal{V}^L$ are provided with labels. For each node $v \in \mathcal{V}^L$, a label vector $\mathbf{y}_v \in \{0,1\}^c$ s.t. $\sum \mathbf{y}_v = 1$ indicates its class membership. $\mathcal{V}^U = \mathcal{V} - \mathcal{V}^L$ is the set of unlabeled nodes whose labels are to be predicted. We use $\mathcal{C}_r \subseteq \mathcal{V}$ to denote an arbitrary cluster of $\mathcal{G}$ containing a subset of nodes and we use $\mathbf{b}_r$ to refer to the label distribution of $\mathcal{C}_r$ which is defined as the average of labels $\mathbf{y}_v$ of all $v \in \mathcal{C}_r$. Finally $\mathbf{Y} \in \{0,1\}^{N \times c}$ is the node label matrix where $\mathbf{y}_v$ is an all-0 vector for any $v \in \mathcal{V}^U$, and $\mathbf{B} \in \mathbb{R}^{C \times c}$ denotes the label distribution for C clusters.

3.1 Graph Neural Networks

GNNs primarily aim to learn vector representations for nodes in a graph through features and topology. The learned representations are used for downstream tasks such as node classification, link prediction, and graph classification. A k-layer GNN consists of k sequential graph convolutional layers that implement *message passing* mechanisms to update representations using aggregated neighborhood nodes representations. The updating process of the k-th layer in GNN is generally formulated as

$$\begin{aligned} \mathbf{h}^k_{\mathcal{N}(v)} &= \text{AGGREGATE}^k(\{\mathbf{h}^{k-1}_u, \forall u \in \mathcal{N}(v)\}), \\ \mathbf{h}^k_v &= \text{UPDATE}^k(\mathbf{h}^k_{\mathcal{N}(v)}; \mathbf{W}^k), \end{aligned} \tag{1}$$

where $\mathbf{h}^k_v$ is node v's representation at the k-th layer, $\mathcal{N}(v)$ is its neighborhood set, and $\mathbf{W}^k$ defines the parameters of the learnable UPDATE function. GCN [20], GraphSAGE [15], and GAT[32] are some of the most widely used GNNs.

3.2 Differential Privacy

Since first proposed by Dwork et al. [11], differential privacy has been established as the de-facto definition for privacy guarantees. In its seminal work, DP provided privacy guarantees for databases in the centralized setting where a trusted curator holds a database containing users' private information and answers queries about the database. In this work, we focus on a decentralized definition of DP referred to as Local Differential Privacy [18]. As opposed to DP which requires a trusted aggregator, LDP provides privacy guarantees on the user side by letting each user perturb their own private data before sending it to an aggregator. This way, the aggregator cannot access the true and private data of any user alleviating the need for a trusted aggregator.

Definition 1. *ϵ-LDP [18]. Given $\epsilon > 0$, a randomized mechanism $\mathcal{M}$ satisfies ϵ-local differential privacy, if for any pairs of user's private data x and x', and for any possible outputs $o \in Range(\mathcal{M})$, we have* $\Pr[\mathcal{M}(x) = o] \leq e^\epsilon \cdot \Pr[\mathcal{M}(x') = o]$*, where ϵ is the privacy budget that controls the trade-off between utility and privacy of $\mathcal{M}$.*

Essentially, LDP ensures that an adversary is unable to infer the input values of any target individual using the output values obtained. To achieve LDP in data statistics and analysis, mechanisms such as randomized response, histogram encoding, unary encoding, or local hashing are applied during the collection of user data that are categorical in nature [33].

3.3 Sampling for Privacy Amplification

A widely accepted approach to strengthen privacy guarantees of DP mechanisms is to take advantage of the aggregator/adversary's uncertainty by adding a sampling step before privatization [21, 4]. Sampling exploits such uncertainty and allows for the relaxation of privacy constraints in a randomization mechanism while ensuring strict privacy guarantees.

Lemma 1. *Amplification Effect of Sampling [21]. Let $\mathcal{M}$ denote an algorithm that guarantees ϵ'-DP over some data. Also, let $\mathcal{M}^{\beta}$ denote an algorithm that first samples tuples from the data with probability β and then applies $\mathcal{M}$ on the sampled data. Then, $\mathcal{M}^{\beta}$ satisfies DP with $\epsilon = \ln\Big(1 + \beta(e^{\epsilon'} - 1)\Big)$.*

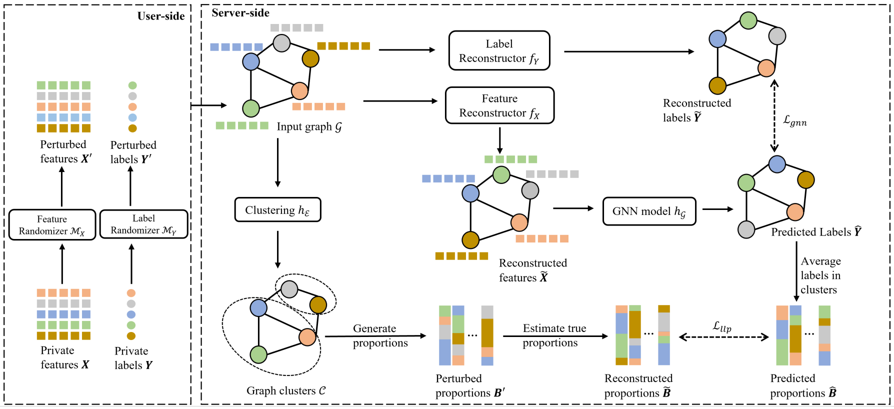
3.4 Learning from Label Proportions

Learning from label proportions (LLP) is an alternative supervision method used to train predictive models when instance labels are too difficult or expensive to obtain [27]. In LLP, instances are grouped into iid bags and only the label distributions of these bags are known to the learning model. The goal is to train an instance label predictor using supervision from bag-level aggregate information.

Denote by $\mathcal{B}_r$, an arbitrary bag containing labeled instances. The learner cannot access the instance labels $\mathbf{y}$ and only receives bag proportions $\mathbf{b}_r$ computed as an average over the instance labels in bag $\mathcal{B}_r$. Then an instance label predictor can simply compute its prediction of the bag proportion $\hat{\mathbf{b}}_r$ by averaging over the predicted labels $\hat{\mathbf{y}}$ in bag $\mathcal{B}_r$. Unlike a traditional learning model, this LLP-based learner calculates training loss between the true and predicted distributions of the bags. For a bag $\mathcal{B}_r$, this proportion loss can be computed as the KL divergence between true and predicted proportions as $D_{KL}(\hat{\mathbf{b}}_r||\mathbf{b}_r)$. The objective of the learner is to minimize $D_{KL}(\cdot)$ for all bags in the training dataset. It has been previously shown that minimizing bag proportion prediction error guarantees a good instance label predictor assuming that the labels are not evenly distributed in all bags [39].

4 Reconstruction-based Private GNN

We formally define the problem of node LDP. Let $\mathcal{M}$ denote some randomization mechanism that provides ϵ-LDP. Then, $\mathbf{X}' = \mathcal{M}(\mathbf{X})$ and $\mathbf{Y}' = \mathcal{M}(\mathbf{Y})$ refer to features and labels collected using $\mathcal{M}$ to ensure user privacy. Let $h(\cdot)$ define a GNN that takes node features and adjacency matrix as the input and outputs label predictions. In this decentralized DP setting, the server responsible for model training has access to the non-private adjacency $\mathbf{A}$, randomized features $\mathbf{X}'$ and randomized labels $\mathbf{Y}'$ but not the true private features $\mathbf{X}$ and labels $\mathbf{Y}$. In this scenario, we aim to train a GNN on $\mathbf{X}'$ and $\mathbf{Y}'$ to learn a mapping $h(\mathbf{X}', \mathbf{A}; \mathbf{W}) \rightarrow \hat{\mathbf{Y}}$ that can estimate accurate labels for $v \in \mathcal{V}^U$.



User-side
Server-side
Perturbed features X'
Perturbed labels Y'
Feature Randomizer $\mathcal{M}_X$
Label Randomizer $\mathcal{M}_Y$
Private features X
Private labels Y
Input graph $\mathcal{G}$
Label Reconstructor f_Y
Feature Reconstructor f_X
Reconstructed labels $\tilde{Y}$
$\mathcal{L}_{gnn}$
Clustering $h_{\mathcal{E}}$
Reconstructed features $\tilde{X}$
GNN model $h_{\mathcal{G}}$
Predicted Labels $\hat{Y}$
Average labels in clusters
Graph clusters $\mathcal{C}$
Generate proportions
Perturbed proportions B'
Estimate true proportions
Reconstructed proportions $\tilde{B}$
$\mathcal{L}_{llp}$
Predicted proportions $\hat{B}$

Figure 1: An overview of the proposed framework RGNN

Fig. 1 illustrates the overall framework of RGNN which is composed of two perturbation models: $\mathcal{M}_X$ and $\mathcal{M}_Y$ implemented at the node level to inject noise into the feature and label data respectively. On the server side, the feature reconstructor f_X takes the non-private graph structure $\mathcal{G}$ and the perturbed features $\mathbf{X}'$ as input to derive estimated features $\tilde{\mathbf{X}}$ via propagation and frequency estimation for all nodes in the graph. The label reconstructor f_Y also takes the graph structure $\mathcal{G}$ and perturbed labels $\mathbf{Y}'$ as input to reconstruct labels for the labeled set of nodes $\mathcal{V}^L$. Additionally, an edge-based graph clustering algorithm $h_{\mathcal{E}}$ partitions the graph into a set of clusters $\mathcal{C}$. For each cluster $\mathcal{C}_r \in \mathcal{C}$, a cluster-level label proportion denoted as $\mathbf{b}'_r$ is computed by aggregating over the nodes in $\mathcal{C}_r$. Utilizing frequency estimation, an accurate cluster-level proportion $\tilde{\mathbf{b}}_r$ is derived from the perturbed cluster label proportion. A GNN model $h_{\mathcal{G}}$ is then trained on the reconstructed features and labels with additional supervision provided by the reconstructed cluster proportions.

4.1 Reconstruction with Private Features

LDP frequency oracles used to obtain private data statistics can also be adopted during training data collection to provide privacy protection to users via plausible deniability. However, in contrast to the one-dimensional data collected for such purpose, data for model training are usually multi-dimensional with mutual dependencies between features. Ensuring formal privacy guarantees then requires users to report privatized values for each feature. In other words, the total privacy budget gets compounded along the feature dimension and results in a high noise rate often at the cost of model performance [33]. To circumvent this issue, sampling and noisy data generation techniques have been adopted in [29, 4] where each user applies randomization techniques only on a few sampled features and reports some default or noisy values for the non-sampled features. The RS+FD method proposed in [4] samples one of d total features and applies an LDP protocol on it. For the remaining $d-1$ features, the method reports completely random values drawn from a uniform distribution over each feature domain. On the other hand, the multi-bit mechanism in [29] randomizes one of d features but reports a default value of 0 for the rest. As discussed prior, the multi-bit mechanism may require high feature matrix sparsity to achieve good utility.

Therefore, in this paper, we focus on sampling and fake data generation using a uniform distribution. Still, in high-dimensional feature settings, reporting $d-1$ features from a uniform distribution may result in highly noisy data unsuitable for training. So, we extend this method to a more general case such that each user samples m of d features and randomizes them independently. Consequently, the privacy budget ϵ is compounded over m sampled features, which is more favorable than compounding over d features. On the client side, a user samples m features then implements the Generalized Randomized Response (GRR) method [34] on the sampled features, i.e.: the user reports their true value with probability p and reports any other value with probability q. Here, p and q are set as $\frac{e^{\epsilon_x}}{e^{\epsilon_x}+\gamma_i-1}$ and $\frac{1}{e^{\epsilon_x}+\gamma_i-1}$ respectively where γ_i indicates the domain size of the sampled feature x and ϵ_x is the allocated privacy budget for feature x. This probabilistic transition can also be represented using a matrix $\mathbf{P} \in \mathbb{R}^{\gamma_i \times \gamma_i}$ whose diagonal entries equal p and the non-diagonal entries equal q. For $d-m$ features, the user reports a random value drawn from a uniform distribution over each feature. We call this method Generalized Randomized Response with Feature Sampling (GRR-FS) which ensures ϵ_X-LDP (Please refer to the **appendix** for proofs).

Theorem 1. *GRR-FS satisfies ϵ_X-LDP where $\epsilon_X = \ln\big(1 + \frac{m}{d}(e^{m\epsilon_x} - 1)\big)$.*

As data randomized via an LDP protocol is invariant to post-processing [12], the server can utilize the collected data without compromising user privacy. Furthermore, as the server queries each user for their feature values only once, privacy degradation via repeated queries is also avoided. GRR-FS can also be applied to continuous node features after discretization. However, directly using noisy features for learning can significantly impact the model's capability of obtaining high-quality embeddings and generalizing to new or unseen data. In this paper, we propose to utilize statistical frequency estimation techniques to approximately reconstruct user features and labels and better facilitate training.

We first derive frequency estimation for GRR-FS. On the server side, the aggregator collects responses from n users but is unaware whether an individual response was derived from GRR or sampled from the uniform distribution. For an arbitrary feature x_i, the aggregator can compute the frequency estimate for its j-th value, denoted by $\tilde{\pi}_j$, as follows

$$\tilde{\pi}_j = \frac{\lambda'_j d}{m(p-q)} + \frac{m - d - m\gamma_i q}{m\gamma_i(p-q)}, \tag{2}$$

where λ'_j denotes the probability of observing the j-th value. The variance of $\tilde{\pi}_j$ is given as

$$\hat{\mathrm{var}}(\tilde{\pi}_j) = \frac{d^2 \tilde{\lambda}_j (1 - \tilde{\lambda}_j)}{m^2 (n-1)(p-q)^2}, \tag{3}$$

where, $\tilde{\lambda}_j = \frac{1}{d}\left(\tilde{\pi}_j m \gamma_i (p-q) + m(q\gamma_i - 1) + d\right)$.

Theorem 2. *For an arbitrary attribute x_i for $i \in \{1, \ldots, d\}$, randomized using GRR-FS with sampling probability m/d, the estimation $\tilde{\pi}_j$ in (2) is an unbiased estimation of the true proportion of users having the j-th value with estimated variance $\hat{\mathrm{var}}(\tilde{\pi}_j)$ in (3).*

For an arbitrary feature x_i, (2) gives us an unbiased estimate of the proportion of nodes with the j-th value over the whole graph. Pointwise reconstruction, however, is still intractable. Nonetheless, we can estimate proportions $\tilde{\boldsymbol{\pi}}$ at a sub-graph level for a reasonable number of nodes, i.e.: for a sub-graph with n nodes, λ'_j is the observed proportion of x_i's j-th value in the sub-graph. We extend this sub-graph estimation to node neighborhoods. For a node $v \in \mathcal{V}$, we first obtain $\boldsymbol{\lambda}'_v$, the proportional frequency of an arbitrary feature x_i from its neighborhood. Using (2), we estimate v's true neighborhood feature distribution as $\tilde{\pi}_{vj}$ for the j-th value in the domain of x_i. Furthermore, considering homophily in graphs, we reason that the feature distribution of a node should be very close to the feature distribution in its neighborhood. Ultimately, we obtain the highest probable feature value from the reconstructed neighborhood feature distribution $\tilde{\pi}_v$ and assign it as v's reconstructed feature $\tilde{x}$. In a simpler case with binary feature values, we can assign $\tilde{\pi}_{v2}$ ($\tilde{\pi}_{v1}$) to be node v's reconstructed feature to incorporate the uncertainty of the estimates.

That said, node degrees in real-world graphs generally follow a power law distribution resulting in a high number of nodes having low degrees [5]. This variation in node degrees affects the reconstruction process as the neighborhood size directly influences the variance in estimating the true feature distribution. In (3), for a fixed privacy budget ϵ_x, we obtain a fixed transition probability p; the variance is then inversely proportional to n which refers to the neighborhood size. This results in low-degree nodes having a higher variance in their estimates which may lead to inaccurate reconstruction. To minimize this effect, we implement multi-hop feature aggregation to increase the neighborhood size for low-degree nodes before computing the estimate.

4.2 Reconstruction with Private Labels

In this section, we discuss the privatization and reconstruction of node labels. We implement GRR to add class-independent and symmetric noise to the labels. Here, the probabilities p and q are set as $\frac{e^{\epsilon_y}}{e^{\epsilon_y}+c-1}$ and $\frac{1}{e^{\epsilon_y}+c-1}$ respectively where c indicates the number of classes and ϵ_y is the allocated label privacy budget. The server collects perturbed label vectors $\mathbf{y}'_v$ from all labeled nodes and computes graph-level aggregates from the collected data. We can further leverage frequency estimation to obtain unbiased estimates of these aggregates. Let $\boldsymbol{\pi} \in \mathbb{R}^c$ denote the label distribution vector over n nodes and $\boldsymbol{\lambda}'$ denote the label distribution observed by the server containing the sample proportions corresponding to $\boldsymbol{\pi}$. Then an unbiased estimate of $\boldsymbol{\pi}$ is obtained as [34]

$$\tilde{\boldsymbol{\pi}} = \mathbf{P}^{-1}\boldsymbol{\lambda}', \tag{4}$$

where $\mathbf{P}$ refers to the label transition matrix with diagonal entries p and non-diagonal entries q. The variance in estimating $\tilde{\boldsymbol{\pi}}$ is obtained from the diagonal elements of the dispersion matrix $\mathrm{disp}(\tilde{\boldsymbol{\pi}}) = (n-1)^{-1}\mathbf{P}^{-1}(\boldsymbol{\lambda}'^{\delta} - \boldsymbol{\lambda}'\boldsymbol{\lambda}'^{\intercal})(\mathbf{P}^{\intercal})^{-1}$ where $\boldsymbol{\lambda}'^{\delta}$ is a diagonal matrix with the same diagonal elements as those of $\boldsymbol{\lambda}'$ and $(\cdot)^{\intercal}$ indicates the transpose operation.

Similar to node features, we can obtain frequency estimates of labels at a sub-graph level using (4). Unlike node features, node labels are only provided for a small subset of nodes in the training graph. Direct propagation over all nodes is not feasible in such graphs. However, unlabeled nodes are also important for message propagation during multi-hop aggregation. To this end, we perform masked propagation to obtain label frequencies in multi-hop node neighborhoods. We mask the unlabeled nodes by setting their label vectors as an all-0 vector of size c. We then use the obtained neighborhood-level label distribution $\boldsymbol{\lambda}'_v$ to estimate the true label distribution $\tilde{\boldsymbol{\pi}}_v$ using (4). Assuming homophily, we assign the reconstructed neighborhood label distribution as the node's new label $\tilde{\mathbf{y}}_v$ after one-hot encoding such that $\tilde{\mathbf{y}}_v \in \{0,1\}^c$. With the reconstructed labels, we define the GNN objective as $\mathcal{L}_{gnn} = \sum_{v \in \mathcal{V}^L} \ell(\hat{\mathbf{y}}_v, \tilde{\mathbf{y}}_v)$, where $\ell(\cdot)$ is the cross-entropy loss, $\tilde{\mathbf{y}}_v$ and $\hat{\mathbf{y}}_v$ denote the reconstructed and predicted label of node v, respectively.

Algorithm 1: Reconstruction based Private GNN (RGNN)

```
Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, features $\mathbf{X}$, labels $\mathbf{Y}$, feature privacy budget $\epsilon_x$, label privacy budget
       $\epsilon_y$, feature randomizer $\mathcal{M}_X$, label randomizer $\mathcal{M}_Y$, feature reconstructor $f_X$, label
       reconstructor $f_Y$, hops for feature reconstruction $K_X$, hops for label reconstruction $K_Y$,
       clustering algorithm $h_{\mathcal{E}}$, number of clusters $C$, GNN model $h_{\mathcal{G}}$, regularization parameter
       $\alpha$
   // randomization
1  $\mathbf{x}'_v \leftarrow \mathcal{M}_X(\mathbf{x}_v; \epsilon_x) \;\; \forall v \in \mathcal{V}$
2  $\mathbf{y}'_v \leftarrow \mathcal{M}_Y(\mathbf{y}_v; \epsilon_y) \;\; \forall v \in \mathcal{V}^L$
   // clustering
3  $\{\mathcal{C}_r\}_{r=1}^{C} \leftarrow h_{\mathcal{E}}(\mathcal{G})$ such that $\mathcal{C}_r \bigcap \mathcal{C}_s = \emptyset$
4  $\mathbf{b}'_r \leftarrow \frac{1}{|\mathcal{C}_r^L|} \sum_{v \in \mathcal{C}_r^L} \mathbf{y}'_v$ for $r = \{1, .., C\}$
   // reconstruction
5  $\tilde{\mathbf{X}} \leftarrow f_X(\mathcal{G}, \mathbf{X}', \gamma, \epsilon_x, d, m, K_x)$
6  $\tilde{\mathbf{Y}}^L \leftarrow f_Y(\mathcal{G}, \mathbf{Y}', \epsilon_y, c, K_y)$
7  obtain $\tilde{\mathbf{b}}_r$ from $\mathbf{b}'_r$ for $r = \{1, \ldots, C\}$ using (4)
   // GNN training
8  for $t = 1, \ldots, T$ do
9      $\hat{\mathbf{Y}} \leftarrow h_{\mathcal{G}}(\tilde{\mathbf{X}}, \mathcal{G}; \mathbf{W})$
10     $\hat{\mathbf{b}}_r \leftarrow \frac{1}{|\mathcal{C}_r^L|} \sum_{v \in \mathcal{C}_r^L} \hat{\mathbf{y}}_v$ for $r = \{1, \ldots, C\}$
11     $\mathcal{L}_{gnn} \leftarrow \frac{1}{|\mathcal{V}^L|} \sum_{v \in \mathcal{V}^L} \ell(\hat{\mathbf{y}}_v, \tilde{\mathbf{y}}_v)$
12     $\mathcal{L}_{llp} \leftarrow \frac{1}{C} \sum_{r=1}^{C} D_{KL}(\hat{\mathbf{b}}_r || \tilde{\mathbf{b}}_r)$
13     $\mathbf{W}^{t+1} \leftarrow \mathbf{W}^t - \eta \nabla \left(\mathcal{L}_{gnn} + \alpha \mathcal{L}_{llp}\right)$
14 end for
```

4.3 Reconstruction with Label Proportions

The variance in node label estimation depends on neighborhood size which could be small for some nodes even after multi-hop aggregation. To alleviate this dependence on neighborhoods, we further introduce a reconstruction-based regularization that incorporates LLP. This LLP regularization term enforces similarity constraints on bag-level label distributions instead of node-level label distributions. We introduce this LLP objective during training by creating bags that contain subsets of nodes in the graph. To this end, we utilize edge-based graph clustering algorithms so as to not consume any privacy budget and partition $\mathcal{G}$ into C disjoint clusters, $\mathcal{C}_1, \ldots, \mathcal{C}_C$. We use METIS [17] to partition the graph in this work. METIS is a multilevel C-way graph partitioning scheme that partitions $\mathcal{G}$ containing N nodes into C disjoint clusters such that each cluster contains around N/C nodes and the number of inter-cluster edges is minimized. In the presence of homophily, nodes in the same cluster most likely share similar labels due to their proximity in $\mathcal{G}$. LLP on bags containing similarly labeled nodes approximates the standard supervised learning and is preferable for training [39].

For each bag/cluster $\mathcal{C}_i$, we compute its label proportion as the mean of the label distribution of nodes in $\mathcal{C}_i$. However, due to privacy constraints, the server cannot access true labels $\mathbf{y}_v$. So, we use randomized labels $\mathbf{y}'_v$ to obtain the perturbed bag proportions $\mathbf{b}'_i$ for each $\mathcal{C}_i$. Note that we only use the nodes that are also in $\mathcal{V}^L$ (denoted as $\mathcal{C}_i^L$ in Line 10 of Algorithm 1) to obtain label proportions of $\mathcal{C}_i$. Since $\mathbf{b}'_i$ is an observed estimate of label proportion at a sub-graph level, we can obtain an unbiased estimate of the bag label proportion using frequency estimation. We formulate the LLP-based objective as $\min_{\mathbf{W}} \mathcal{L}_{llp} = \sum_{i=1}^{C} D_{KL}(\hat{\mathbf{b}}_i || \tilde{\mathbf{b}}_i)$, where $\hat{\mathbf{b}}_i$ is the predicted label proportion of bag i obtained by aggregating the predicted labels $\hat{\mathbf{y}}_v$ of nodes in $\mathcal{C}_i^L$, $\tilde{\mathbf{b}}_i$ is the reconstructed label proportion of bag i obtained using (4). Also, $\mathbf{W}$ denotes the learnable parameters of a GNN model $h_{\mathcal{G}}(\cdot)$. The overall GNN objective with LLP regularization follows as

$$\min_{\mathbf{W}} \mathcal{L}_{gnn} + \alpha \mathcal{L}_{llp}, \tag{5}$$

where α controls the influence of the LLP-based regularization. The overall training procedure of our model is presented in Algorithm 1 (Complete pseudocodes for reconstruction components f_X and f_Y are included in the **appendix**). On the user side, randomization is performed on both features and labels (Lines 1-2). The server has access to graph adjacency and performs a clustering operation on it to obtain clusters to be used as bags for LLP (Lines 3-4). The server uses the perturbed data along with the graph adjacency to estimate node features and labels, and bag proportions via reconstruction as discussed prior (Lines 5-7). Finally, the server trains a GNN model to fit the reconstructed features and labels with the LLP loss as regularization (Lines 9-13).

Theorem 3. *Algorithm 1 satisfies* $(\epsilon_X + \epsilon_y)$*-LDP where* $\epsilon_X = \ln\big(1 + \frac{m}{d}(e^{m\epsilon_x} - 1)\big)$.

5 Experiments

5.1 Experimental Setup

We evaluate RGNN on four real-world datasets: Citeseer, Cora, DBLP, Facebook and two semi-synthetic datasets: German and Student. Detailed information about the datasets, semi-synthetic dataset construction, and preprocessing to reduce feature sparsity are presented in the **appendix**. For all datasets, we compare RGNN against a stand-alone GNN model trained directly on the randomized features and labels. We demonstrate RGNN's efficiency by comparing it against LPGNN [29] which performs node classification in the same node privacy scenario. For LPGNN, we use the multi-bit mechanism to perturb features and RR to perturb labels as discussed in [29].

We implement a two-layer GNN model with 16 hidden dimensions. We use GraphSAGE as the base GNN model unless stated otherwise. For all models, we use ReLU [3] as the non-linear activation followed by dropout and train with Adam [19] optimizer. We vary the propagation parameters K_x and K_y among {2, 4, 8, 16}. For a fair comparison, we use the same propagation parameters in both RGNN and LPGNN for each dataset. We further vary the hyperparameters C among {4, 8, 16, 32, 64, 128, 256} and α among {0.01, 0.1, 1, 10, 20}. Detailed experiments on the effects of the propagation and regularization parameters and the choice of the base GNN model can be found in the appendix. To study the performance of RGNN under varying privacy budgets, we vary ϵ_x within {1, 0.1, 0.01} and ϵ_y within {3, 2, 1, 0.5} and fix m=10. The total feature privacy budget provided by GRR-FS, ϵ_X, is then computed as in Theorem 1. For instance, for the Citeseer dataset the corresponding values for ϵ_X varies within {8.3, 0.3, 0.02}. Accordingly, we set LPGNN's sampling parameter to be m. Since LPGNN also performs sampling before randomization, we reason that the total feature privacy for LPGNN is also amplified resulting in ϵ_X feature privacy. For all datasets, we randomly split nodes into training, validation, and test sets with 50/25/25% ratios, respectively. We report the average results with standard deviations of 5 runs trained for 100 epochs each for all experiments. All models are implemented using PyTorch-Geometric [14] and are run on GPU Tesla V100 (32GB RAM). Our implementation is available at `https://github.com/karuna-bhaila/RGNN`.

5.2 Comparison with Baselines

We evaluate all models on the semi-synthetic and the pre-processed real-world datasets with reduced feature sparsity and report the results in Table 1 under $(\epsilon_X + \epsilon_y)$-LDP. Here, GraphSAGE refers to the stand-alone GNN that directly uses the perturbed features and labels. We observe that RGNN significantly improves performance with some utility trade-off for all datasets. For semi-synthetic Student dataset, RGNN can achieve better or similar accuracy compared to the non-private baseline for higher label privacy budgets (Please refer to Table 2 in appendix for the performance of the non-private GNN). This can be attributed to its size and the higher degree of feature homophily present in the graph owing to the nature of its construction. Generally, RGNN also achieves significantly better accuracy compared to LPGNN. Out of 48 scenarios, RGNN outperforms LPGNN for 39 of them. The difference in accuracy is mostly minimal otherwise. As discussed previously, the multi-bit mechanism in LPGNN preserves the sparsity of the feature matrix when $m < d$. Furthermore, even after pre-processing to reduce dimensions, the datasets are not evenly distributed in terms of the feature domains. Compared with this, RGNN has comparable or improved performance despite providing rigorous privacy protection by randomizing every feature. These results imply that the reconstruction-based framework is effective in improving model performance under LDP constraints.

Table 1: Accuracy of RGNN and baselines under $(\epsilon_X + \epsilon_y)$-LDP with $m = 10$

Dataset	ϵ_x	Model	$\epsilon_y = 3$	$\epsilon_y = 2$	$\epsilon_y = 1$	$\epsilon_y = 0.5$
Citeseer	1	GraphSAGE	24.4 ± 1.3	24.3 ± 2.6	20.5 ± 1.2	19.4 ± 0.9
		LPGNN	45.6 ± 12.2	51.7 ± 1.8	46.4 ± 1.6	26.9 ± 5.9
		RGNN	**55.7 ± 1.4**	**52.6 ± 2.7**	**47.1 ± 2.8**	**36.2 ± 4.7**
	0.1	GraphSAGE	25.2 ± 1.3	24.6 ± 1.1	19.4 ± 1.2	19.4 ± 1.2
		LPGNN	**53.1 ± 1.9**	**51.3 ± 3.4**	**47.3 ± 1.8**	31.1 ± 6.3
		RGNN	51.7 ± 2.1	**51.3 ± 2.3**	45.1 ± 3.2	**34.6 ± 3.0**
Cora	1	GraphSAGE	31.5 ± 1.9	28.0 ± 1.6	25.9 ± 2.4	20.4 ± 4.4
		LPGNN	55.5 ± 15.4	39.5 ± 14.2	34.1 ± 8.4	37.8 ± 14.2
		RGNN	**77.8 ± 2.0**	**75.5 ± 1.6**	**67.5 ± 3.8**	**41.9 ± 3.0**
	0.1	GraphSAGE	32.2 ± 1.9	28.5 ± 0.8	24.8 ± 2.4	21.0 ± 3.3
		LPGNN	68.5 ± 1.8	63.8 ± 5.4	60.6 ± 3.5	**44.8 ± 13.2**
		RGNN	**77.0 ± 1.5**	**75.8 ± 2.0**	**66.6 ± 5.0**	40.6 ± 2.3
DBLP	1	GraphSAGE	50.3 ± 1.4	50.2 ± 1.5	45.5 ± 0.9	42.4 ± 2.4
		LPGNN	65.2 ± 0.5	65.0 ± 0.4	60.7 ± 6.7	46.6 ± 1.5
		RGNN	**71.2 ± 0.4**	**70.7 ± 0.7**	**70.0 ± 1.0**	**65.5 ± 2.3**
	0.1	GraphSAGE	49.1 ± 1.7	48.7 ± 2.4	44.5 ± 1.7	43.8 ± 1.4
		LPGNN	70.2 ± 1.1	68.5 ± 1.8	64.6 ± 4.4	59.9 ± 3.0
		RGNN	**71.4 ± 0.7**	**71.0 ± 0.8**	**69.9 ± 0.9**	**65.8 ± 2.0**
Facebook	1	GraphSAGE	46.4 ± 1.3	45.2 ± 1.3	35.5 ± 1.1	29.7 ± 1.4
		LPGNN	70.4 ± 2.1	69.3 ± 1.7	67.6 ± 1.9	58.7 ± 10.8
		RGNN	**76.3 ± 0.9**	**75.9 ± 1.1**	**72.6 ± 1.6**	**64.5 ± 1.6**
	0.1	GraphSAGE	42.0 ± 1.3	41.4 ± 1.5	33.5 ± 1.2	29.6 ± 1.3
		LPGNN	76.3 ± 0.9	75.8 ± 0.8	73.1 ± 0.3	**67.4 ± 1.3**
		RGNN	**76.5 ± 0.8**	**76.1 ± 1.2**	**73.2 ± 1.2**	63.4 ± 1.1
German	1	GraphSAGE	76.8 ± 3.1	73.1 ± 4.3	70.9 ± 2.8	70.1 ± 4.2
		LPGNN	69.6 ± 1.5	69.6 ± 1.5	69.6 ± 1.5	69.6 ± 1.5
		RGNN	**82.2 ± 3.5**	**82.3 ± 3.5**	**82.2 ± 6.2**	**83.1 ± 7.9**
	0.1	GraphSAGE	74.1 ± 5.2	72.3 ± 2.9	70.8 ± 2.6	69.3 ± 6.3
		LPGNN	74.7 ± 5.0	73.1 ± 7.0	71.3 ± 6.7	72.8 ± 5.
		RGNN	**81.9 ± 4.4**	**81.9 ± 4.2**	**82.9 ± 6.9**	**83.2 ± 7.1**
Student	1	GraphSAGE	73.6 ± 4.5	68.9 ± 3.1	63.8 ± 7.8	56.2 ± 3.8
		LPGNN	**90.3 ± 2.4**	**89.9 ± 2.0**	85.1 ± 4.1	76.7 ± 9.7
		RGNN	88.9 ± 1.6	87.8 ± 1.2	**88.2 ± 0.9**	**82.1 ± 3.6**
	0.1	GraphSAGE	73.2 ± 4.7	68.8 ± 4.9	62.2 ± 6.7	56.9 ± 3.6
		LPGNN	88.1 ± 1.9	**89.7 ± 1.9**	**87.4 ± 2.2**	78.1 ± 9.7
		RGNN	**88.5 ± 1.7**	88.5 ± 1.1	**87.4 ± 1.5**	**81.7 ± 4.2**

5.3 Ablation Study

We conduct an ablation study to evaluate the contribution of the reconstruction and LLP components. To this end, we train three variants of RGNN: RGNN$\backslash f_X$ uses the perturbed features directly in training, RGNN$\backslash f_Y$ uses perturbed labels for training, and RGNN$\backslash$LLP without LLP regularization. With $m = 10$, $\epsilon_x = 1$ and $\epsilon_y = 0.5$, we obtained the following results: (**40.2±1.9**, 34.0±3.2, 28.3±6.6, 23.6±1.5) for Citeseer and (**66.2±1.4**, 63.0±1.0, 64.9±1.0, 46.1±1.3) for DBLP with RGNN, and RGNN$\backslash$LLP, RGNN$\backslash f_Y$, RGNN$\backslash f_X$ respectively. We observe that the performance of RGNN$\backslash f_X$ is significantly worse than that of RGNN, which highlights the contribution of denoising node features via frequency estimation. The performance of RGNN is also better than that of RGNN$\backslash f_Y$ and RGNN$\backslash$LLP, showing that label reconstruction and reconstructed LLP improves model performance.

6 Conclusion

We present a reconstruction-based learning framework for GNNs with private features and labels. We derive a perturbation mechanism with sampling that is implemented at the user level and ensures privacy via plausible deniability. We then propose a flexible training framework that can significantly improve the privacy-utility tradeoff of the learner. The proposed method utilizes statistical frequency estimation to approximate true node features and labels. To reduce estimation variance, we incorporate a simple propagation mechanism that aggregates information from multi-hop neighborhoods. We also introduce a regularization technique that uses label proportions of graph clusters to supervise the learning process at a sub-graph level. Our experiments demonstrate that our method generalizes well across various datasets and GNN architectures while providing rigorous privacy guarantees.

Acknowledgements

This work was supported in part by NSF 1946391 and 2119691.

References

[1] Apple's 'differential privacy' is about collecting your data but not your data. `https://www.wired.com/2016/06/apples-differential-privacy-collecting-data/`.

[2] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS*, 2016.

[3] Abien Fred Agarap. Deep learning using rectified linear units (ReLU), 2019.

[4] Héber Hwang Arcolezi, Jean-François Couchot, Bechara al Bouna, and Xiaokui Xiao. Random sampling plus fake data: Multidimensional frequency estimates with local differential privacy. In *CIKM*, 2021.

[5] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 1999.

[6] Karuna Bhaila, Yongkai Wu, and Xintao Wu. Fair collective classification in networked data. In *Big Data*, 2022.

[7] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *ICLR*, 2018.

[8] Mahawaga Arachchige Pathum Chamikara, Peter Bertók, Ibrahim Khalil, Dongxi Liu, Seyit Camtepe, and Mohammed Atiquzzaman. Local differential privacy for deep learning. *IEEE Internet Things J.*, 2020.

[9] Ameya Daigavane, Gagan Madan, Aditya Sinha, Abhradeep Guha Thakurta, Gaurav Aggarwal, and Prateek Jain. Node-level differentially private graph neural networks, 2022.

[10] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. Collecting telemetry data privately. In *NeurIPS*, 2017.

[11] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[12] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 2014.

[13] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: randomized aggregatable privacy-preserving ordinal response. In *CCS*, 2014.

[14] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric, 2019.

[15] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[16] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing links from graph neural networks. In *USENIX Security Symposium*, 2021.

[17] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 1998.

[18] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What can we learn privately? In *FOCS*, 2008.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[20] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[21] Ninghui Li, Wahbeh H. Qardaji, and Dong Su. On sampling, anonymization, and differential privacy or, *k*-anonymization meets differential privacy. In *ASIACCS*, 2012.

[22] NIST. Privacy framework. `https://www.nist.gov/privacy-framework/privacy-framework`.

[23] Iyiola E. Olatunji, Thorben Funke, and Megha Khosla. Releasing graph neural networks with differential privacy guarantees, 2021.

[24] Iyiola E. Olatunji, Wolfgang Nejdl, and Megha Khosla. Membership inference attack on graph neural networks. In *TPS-ISA*, 2021.

[25] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. Scalable private learning with PATE. In *ICLR*, 2018.

[26] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, 2017.

[27] Novi Quadrianto, Alexander J. Smola, Tibério S. Caetano, and Quoc V. Le. Estimating labels from label proportions. *JMLR*, 2009.

[28] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-Scale Attributed Node Embedding. *Journal of Complex Networks*, 2021.

[29] Sina Sajadmanesh and Daniel Gatica-Perez. Locally private graph neural networks. In *CCS*, 2021.

[30] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 2008.

[31] Khang Tran, Phung Lai, NhatHai Phan, Issa Khalil, Yao Ma, Abdallah Khreishah, My T. Thai, and Xintao Wu. Heterogeneous randomized response for differential privacy in graph neural networks. In *Big Data*, 2022.

[32] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[33] Teng Wang, Xuefeng Zhang, Jingyu Feng, and Xinyu Yang. A comprehensive survey on local differential privacy toward data statistics and analysis. *Sensors*, 2020.

[34] Yue Wang, Xintao Wu, and Donghui Hu. Using randomized response for differential privacy preserving data collection. In *EDBT/ICDT Workshops*, 2016.

[35] Stanley L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 1965.

[36] Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. LINKTELLER: recovering private edges from graph neural networks via influence analysis. In *IEEE S&P*, 2022.

[37] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 2021.

[38] Mengmeng Yang, Lingjuan Lyu, Jun Zhao, Tianqing Zhu, and Kwok-Yan Lam. Local differential privacy and its applications: A comprehensive survey. *arXiv preprint arXiv:2008.03686*, 2020.

[39] Felix X. Yu, Krzysztof Choromanski, Sanjiv Kumar, Tony Jebara, and Shih-Fu Chang. On learning from label proportions, 2015.

[40] Qiuchen Zhang, Jing Ma, Jian Lou, Carl Yang, and Li Xiong. Towards training graph neural networks with node-level differential privacy, 2022.

[41] Elena Zheleva and Lise Getoor. Preserving the privacy of sensitive relationships in graph data. In *Privacy, Security, and Trust in KDD*. Springer, 2007.

A Proof of Theorem 1

Proof. Consider a feature vector $\mathbf{x} = \{x_1, \dots, x_d\}$ on which GRR-FS is applied with feature sampling probability $\beta = m/d$ and privacy budget ϵ_x on each sampled feature. Let γ_i refer to the domain size of feature x_i s.t. $x_i \in \{1, \dots, \gamma_i\}$. The private feature vector $\mathbf{x}' = \{x'_1, \dots, x'_d\}$ is obtained as

$$x'_i = \begin{cases} \mathrm{GRR}\,(\epsilon_x, x_i) & \text{with probability } \beta \\ \mathrm{Unif}(1, \gamma_i) & \text{with probability } 1 - \beta, \end{cases} \tag{6}$$

where $\mathrm{GRR}(\cdot)$ denotes the response obtained by applying randomized response and $\mathrm{Unif}(\cdot)$ denotes a sample drawn from a discrete uniform distribution. For each of the m sampled features, the sampling and reporting together satisfy ϵ_x-LDP, and for each of the $d - m$ features, the reporting ensures total randomness. Since ϵ-LDP composes [12], the mechanism composes as $m \cdot \epsilon_x$. Additionally, since we incorporate sampling at the rate of β for each user, we are able to leverage privacy amplification, enabled by the combination of sampling with a randomization mechanism as shown in Lemma 1. Therefore, GRR-FS, with sampling rate β and applied privacy budget ϵ_x for each of m sampled features, satisfies ϵ_X-LDP with $\epsilon_X = \ln\left(1 + \frac{m}{d}(e^{m\epsilon_x} - 1)\right)$. □

B Proof of Theorem 2

Proof. For an arbitrary feature x_i, let π_j denote the true proportion of users who have the j-th value for $1 \le j \le \gamma_i$. Under GRR-FS, the probability of observing the j-th value is

$$\lambda_j = \frac{1}{n}\left[\frac{m}{d}(n\pi_j p + n(1 - \pi_j)q_2) + \left(1 - \frac{m}{d}\right)\frac{n}{\gamma_i}\right] \tag{7}$$

$$= \frac{1}{d\gamma_i}\left[m\pi_j\gamma_i(p - q) + mq\gamma_i + d - m\right]. \tag{8}$$

An unbiased estimator of λ_j is the observed proportion of the j-value, $\lambda'_j = n_j/n$, where n_j refers to the number of users who report the j-th value. Then from (8), it follows

$$\tilde{\pi}_j = \frac{\lambda'_j d}{m(p - q)} + \frac{m - d - m\gamma_i q}{m\gamma_i(p - q)^2}. \tag{9}$$

From (9), we have

$$\mathrm{var}(\tilde{\pi}_j) = \frac{d^2\,\mathrm{var}(n_j)}{n^2 m^2 (p - q)^2}. \tag{10}$$

We observe that n_j follows the binomial distribution with parameters n and λ_j such that $\mathrm{var}(n_j) = n\lambda_j(1 - \lambda_j)$ and we get

$$\mathrm{var}(\tilde{\pi}_j) = \frac{d^2\lambda_j(1 - \lambda_j)}{nm^2(p - q)^2}. \tag{11}$$

Since $\mathbb{E}(n_j) = n\lambda_j$ and $\mathbb{E}(n_j^2) = n\lambda_j + n(n - 1)\lambda_j$, we have

$$\mathbb{E}\left[\frac{\tilde{\lambda}_j(1 - \tilde{\lambda}_j)}{n - 1}\right] = \frac{\lambda_j(1 - \lambda)}{n}, \tag{12}$$

where we use $\tilde{\lambda}_j$ to mean the estimated probability of observing the j-th value obtained as $\tilde{\lambda}_j = \frac{1}{d}\left(\tilde{\pi}_j m\gamma_i(p - q) + m(q\gamma_i - 1) + d\right)$. Finally, an estimation of the variance in (11) is given as

$$\hat{\mathrm{var}}(\tilde{\pi}_j) = \frac{d^2\tilde{\lambda}_j(1 - \tilde{\lambda}_j)}{m^2(n - 1)(p - q)^2}. \tag{13}$$

□

C Proof of Theorem 3

Proof. We show in Theorem 1 that GRR-FS provides ϵ_X-LDP. Also, the label randomization mechanism $\mathcal{M}_Y$ which implements GRR ensures ϵ_y-LDP. According to the compositional property of DP [12], the randomized feature and label data collectively provide $(\epsilon_X + \epsilon_y)$-LDP. The server collects query responses from each node independently only once, thus maintaining the privacy guarantee. Finally, due to the invariance of randomization mechanisms to post-processing [12], privacy protection is not compromised during any step in the reconstruction or training process. Therefore, Algorithm 1 satisfies $(\epsilon_X + \epsilon_y)$-LDP. □

D More Details on Feature and Label Reconstruction

The algorithm to obtain reconstructed node features by the server from the perturbed ones is shown in Algorithm 2.

```
**Algorithm 2:** Node Feature Reconstruction
**Input:** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, perturbed features $\mathbf{X}'$, feature domain sizes $\gamma = \{\gamma_1, \dots, \gamma_d\}$, number of
        features $d$, number of sampled features $m$, privacy budget $\epsilon$, number of hops $K$
**Output:** Reconstructed features $\tilde{\mathbf{X}}$
**Function** $f_X(\mathcal{G}, \mathbf{X}', \gamma, \epsilon, d, m, K)$:
    **for** $i = 1, \dots, d$ **do**
        $\boldsymbol{\lambda}' \leftarrow \text{one-hot}(\mathbf{X}'_i)$
        **for** $k = 1, \dots, K$ **do**
            **for** $v \in \mathcal{V}$ **do**
                $\boldsymbol{\lambda}'_v \leftarrow \text{MEAN}(\{\boldsymbol{\lambda}'_v\} \cup \{\boldsymbol{\lambda}'_u, \forall u \in \mathcal{N}(v)\})$
            **end for**
        **end for**
        $p \leftarrow \frac{e^\epsilon}{e^\epsilon + \gamma_i - 1}$ and $q \leftarrow \frac{1}{e^\epsilon + \gamma_i - 1}$
        **for** $v \in \mathcal{V}$ **do**
            **for** $j = 1, \dots, \gamma_i$ **do**
                $\tilde{\pi}_{vj} \leftarrow \frac{d\lambda'_{vj}}{m(p-q)} + \frac{m-d-m\gamma_i q}{m\gamma_i(p-q)}$
            **end for**
            $\tilde{x}_v \leftarrow \operatorname*{argmax}_j \tilde{\pi}_{vj}$
        **end for**
        $\tilde{\mathbf{x}}_i \leftarrow \{\tilde{x}_v, \forall v \in \mathcal{V}\}$
    **end for**
    **return** $\tilde{\mathbf{X}} = \{\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_d\}^\intercal$
**end**
```

Algorithm 3 shows the process of approximating labels through frequency estimation using perturbed labels.

E Datasets Details

We perform evaluations on four real-world benchmark datasets: Citeseer [30], Cora [30] and DBLP [7] are well-known citation datasets where nodes represent papers and edges denote citations. Each node is described by bag-of-words features and a label denoting its category. Facebook [28] is a social network dataset with verified Facebook sites as nodes and mutual likes as links. Node features represent site descriptions and label indicates its category. Statistics of the datasets are presented in Table 2. Sparsity in d highlights the imbalanced distribution of binary feature values as we discussed in Section 1. To reduce such feature matrix sparsity, we preprocess these real-world datasets by combining a fixed number of features (70, 25, 50, and 100 for Citeseer, Cora, DBLP, and Facebook respectively) into one representative feature resulting in a lower feature dimension

Algorithm 3: Node Label Reconstruction

```
Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, perturbed labels $\mathbf{Y}'$, number of classes $c$, privacy budget $\epsilon$, number of hops $K$
Output: Reconstructed labels $\tilde{\mathbf{y}}_v \ \forall v \in \mathcal{V}^L$
Function $f_Y(\mathcal{G}, \mathbf{Y}', \epsilon, c, K)$:
    $\boldsymbol{\lambda}'_v \leftarrow \mathbf{y}'_v \ \forall v \in \mathcal{V}^L$
    $\boldsymbol{\lambda}'_v \leftarrow \vec{0} \quad \forall v \in \mathcal{V}^U$
    for $k = 1, \ldots, K$ do
        for $v \in \mathcal{V}$ do
            $\boldsymbol{\lambda}'_v \leftarrow \mathrm{MEAN}(\{\boldsymbol{\lambda}'_v\} \cup \{\boldsymbol{\lambda}'_u, \forall u \in \mathcal{N}(v)\})$
        end for
    end for
    $p \leftarrow \frac{e^\epsilon}{e^\epsilon + c - 1}$ and $q \leftarrow \frac{1}{e^\epsilon + c - 1}$
    Construct transition matrix $\mathbf{P} \in \mathbb{R}^{c \times c}$ using $p$ and $q$
    for $v \in \mathcal{V}^L$ do
        $\tilde{\boldsymbol{\pi}}_v \leftarrow \mathbf{P}^{-1} \boldsymbol{\lambda}'_v$
        $\tilde{\mathbf{y}}_v \leftarrow$ one-hot$(\tilde{\boldsymbol{\pi}}_v)$
    end for
    return $\tilde{\mathbf{y}}_v \ \forall v \in \mathcal{V}^L$
end
```

Table 2: Dataset statistics

Dataset	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$	Avg. Deg.	d	Sparsity in d(%)	c	d'	Sparsity in d'(%)	Accuracy(%)
Citeseer	3,327	4,552	2.7	3,703	99.2	6	53	56.0	74.7±1.2
Cora	2,708	5,278	3.9	1,433	98.7	7	58	73.8	87.5±0.9
DBLP	17,716	52,867	6.0	1,639	99.7	4	33	85.3	84.7±0.3
Facebook	22,470	170,912	15.2	4,714	99.7	4	48	82.8	94.2±0.3
German	955	9,900	20.9	46	74.2	2	-	-	88.1±4.0
Student	577	4,243	14.7	60	71.1	2	-	-	86.3±3.8

indicated as d' in Table 2. We choose varying numbers of features for aggregation to obtain different levels of sparsity on the datasets as indicated in the *Sparsity in* d' column in Table 2. Additionally, this process only requires the server to communicate the number of features to be grouped to the users and does not affect the privacy guarantees of RGNN.

We also evaluate RGNN on two semi-synthetic datasets, German [6] and Student [6]. In the German dataset, nodes represent clients at a German bank and the label classifies clients as good or bad customers. In Student dataset, nodes represent students at two Portuguese schools and the label indicates their final grade. Both of these datasets contain both numerical and categorical features and we discretize the numerical attributes based on the quantile distribution of their values. Compared to real-world datasets, both German and Student are smaller in scale with relatively balanced distribution in the feature domains. For both datasets, synthetic edges are generated by selecting top-*k* node pairs with the highest feature similarity quantified as the Euclidean distance between features of the pair.

The rightmost column in Table 2 shows node classification accuracies with unperturbed features. We provide this result as the non-private baseline measure of performance on each dataset.

F Additional Experimental results

F.1 Choice of GNN

In Fig. 2, we plot the performance of RGNN that uses GraphSAGE, GCN, and GAT as the backbone GNN. We only report the results on Citeseer as we observed similar trends on other datasets. In the non-private setting (∞, ∞), GAT outperforms both GCN and GraphSAGE. In the private scenario, GAT and GraphSAGE have relatively similar performances and GCN has comparatively lower utility.

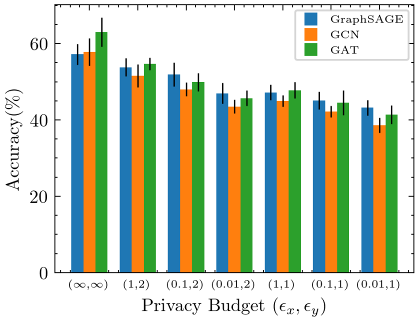


Figure 2: Comparison of GNN architectures

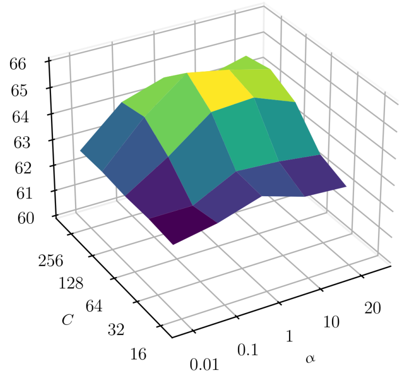


Figure 3: Hyperparameter study of α and C

This trend suggests that the attention mechanism of GAT is able to effectively utilize the reconstructed features to compute attention coefficients. The neighborhood feature aggregator in GraphSAGE also benefits from such reconstruction. The results of this experiment demonstrate the flexibility of the proposed framework in fitting the reconstructed data to popular GNN models with satisfying performance.

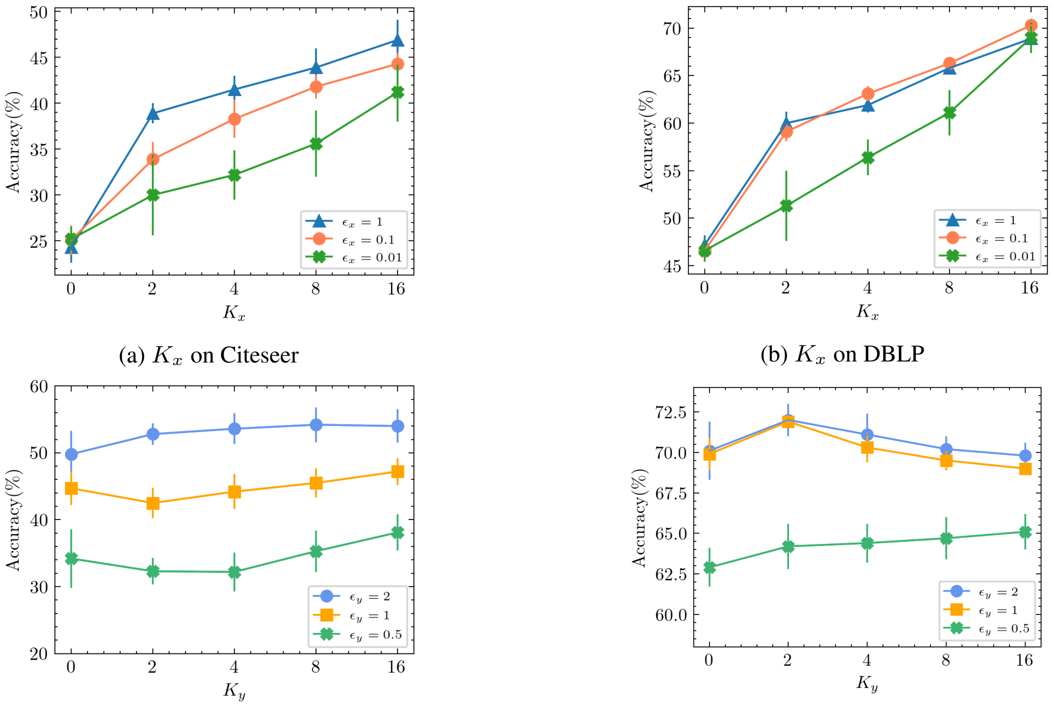
F.2 Hyperparameter Sensitivity

We analyze the effects of hyperparameters in the regularization term $\mathcal{L}_{llp}$. We vary α as $\{0.01, 0.1, 1, 10, 20\}$ and number of clusters C as $\{16, 32, 64, 128, 256\}$ and report the results in Fig. 3 for a fixed privacy budget at $\epsilon_x = 1$ and $\epsilon_y = 0.5$ on the DBLP dataset. We observe that generally as α increases, the performance increases and then decreases with the best performance at $\alpha \in \{0.1, 1, 10\}$. Also, performance increases as C increases and the best results are obtained at $C = 128$ for this dataset. Note that the optimal value of C may vary depending on the size of the dataset which determines the bag size during clustering. The results demonstrate that the learner can benefit from the additional supervision provided by reconstructed LLP at a sub-graph level.

F.3 Propagation Parameter Study

We further investigate the effect of propagation on the reconstruction framework. We vary the propagation parameters K_x and K_y among {0, 2, 4, 8, 16} and report the results in Fig. 4 for different values of ϵ_x and ϵ_y while fixing $m = 10$ on Citeseer and DBLP datasets. We observe that there is a drastic improvement in performance when increasing K_x from 0 to 2 for both datasets. The performance further improves as K_x increases for all values of ϵ_x. This empirically proves that multi-hop propagation is effective in improving the estimates of node features during reconstruction.

Although the performance gain is not as drastic compared to K_x, model utility generally rises as K_y increases. For Citeseer, at lower ϵ_y, larger values of K_y are needed to significantly improve performance. For DBLP, higher values of K_y turn out to be detrimental to model performance for larger ϵ_y but beneficial for small ϵ_y. We speculate that the comparatively higher node degrees in DBLP result in over-smoothing of the labels during multi-hop aggregation when lesser noise is added to them. This observation is also supported by our results on Facebook whose optimal K_x & K_y both turn out to be 4. We generally conclude that RGNN can benefit from long-range propagation, especially with lower privacy budgets, but the selection of these parameters should depend on the desired privacy budget and average node degree of the graph.



(a) K_x on Citeseer

(b) K_x on DBLP

(c) K_y on Citeseer

(d) K_y on DBLP

Figure 4: Influence of propagation parameters K_x and K_y on the performance of RGNN