Structural Differential Privacy in Graph Neural Networks

A thesis submitted in partial fulfillment of the requirements for the award of the degree of

Master of Technology

in

Cryptology and Security

submitted by

Bibek Giri
(Roll No. CrS2305)

Under the esteemed guidance of

Prof. Subhankar Mishra
School of Computer Sciences
NISER Bhubaneswar

Prof. Debrup Chakraborty
Cryptology and Security Research Unit
Indian Statistical Institute, Kolkata

Indian Statistical Institute
Kolkata – 700108, India

July, 2025

Declaration of Authorship

I, **Bibek Giri**, hereby declare that this Master's thesis entitled, ***"Structural Differential Privacy in Graph Neural Networks"***, is the outcome of my research, carried out under the esteemed guidance of **Prof. Subhankar Mishra** and **Prof. Debrup Chakraborty**.

While certain ideas, methods, and insights have been drawn from previously published research and are duly cited in the thesis, the overall work presented here is the result of my own research efforts. To the best of my knowledge, this thesis does not contain any substantial portion of work previously submitted for the award of any degree or diploma at the Indian Statistical Institute or any other academic institution.

I respectfully authorize the Indian Statistical Institute, Kolkata, the right to make this thesis available for academic and research purposes, and to share it with other institutions or individuals for the advancement of scholarly work.

Date: July 11, 2025

Bibek Giri
Roll No. CrS2305
Indian Statistical Institute, Kolkata

Certificate

This is to certify that the dissertation titled ***"Structural Differential Privacy in Graph Neural Networks"*** submitted by **Bibek Giri** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Cryptology and Security** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in our opinion, has reached the standard needed for submission.

Subhankar Mishra

Prof. Subhankar Mishra
Reader-F, School of Computer Sciences
NISER, Bhubaneswar.

Acknowledgements

I am deeply grateful to **Prof. Subhankar Mishra** and **Prof. Debrup Chakraborty** for giving me the opportunity to carry out my Master's thesis under their esteemed guidance. I feel truly fortunate to have learned from their experience and insights throughout this journey. Their patient mentorship, constructive feedback, and encouraging nature have played a vital role in shaping my academic growth and helping me confidently pursue independent research.

I would also like to express my sincere thanks to **Dr. Rucha Bhalchandra Joshi** for her invaluable support and guidance during the course of my dissertation. Her thoughtful inputs and constant encouragement were instrumental in helping me understand the deeper aspects of the subject and improve my work. I greatly appreciate her generous help, especially during challenging phases, and her willingness to assist while allowing me the freedom to explore and learn on my own.

Finally, I would like to express my heartfelt thanks to my family, friends, and batchmates for their unwavering support and encouragement throughout this journey. Their patience, understanding, and constant motivation have been a source of strength for me. This achievement would not have been possible without their belief in me at every step.

Abstract

Graph Neural Networks (GNNs) have demonstrated impressive performance across a range of graph-based learning tasks. However, their application to domains with sensitive relational data raises serious privacy concerns, as the graph structure itself may leak confidential information.

This thesis investigates a decentralized framework for enforcing edge-level *local differential privacy* (LDP) in graph-structured data. We introduce two mechanisms that perturb a node's neighborhood in a privacy-preserving yet utility-aware manner. The first approach replaces randomly selected neighbors with feature-similar nodes from the 2-hop neighborhood, ensuring structural realism while preserving degree. The second approach eliminates the need for explicit 2-hop propagation and dummy vectors, instead relying on randomized feature queries to identify plausible substitutes.

Both approaches are evaluated on benchmark graph datasets such as Cora, PubMed, and LastFM using GNN architectures like GCN, GraphSAGE, and GAT. Experimental results show that our methods achieve a favorable trade-off between structure privacy and learning utility, while avoiding the overhead and privacy leakage risks of centralized or semi-local protocols.

Contents

Chapter 1

Introduction

1.1 Motivation

Graph Neural Networks (GNNs) are powerful deep learning models designed to learn from graph-structured data. Their ability to effectively capture and reason over relational dependencies has led to widespread adoption in various domains, including social networks (e.g., community detection and recommendation), biology and healthcare (e.g., protein interaction prediction and drug discovery), natural language processing (e.g., text classification and machine translation), and transportation systems (e.g., traffic prediction and route optimization).

Despite their broad applicability, the deployment of GNNs in real-world scenarios—especially those involving sensitive or personal data—raises significant privacy concerns. In many settings, such as social networking or messaging platforms, graphs encode sensitive relationships between individuals. Even when the content exchanged across these networks is encrypted, the structural patterns alone—such as who communicates with whom and how frequently—can leak private information. An adversary analyzing such structures may infer friendships, identify community affiliations, or even uncover political or ideological associations.

To mitigate these risks, privacy-preserving mechanisms must be integrated into the learning process. A promising solution is *Differential Privacy* (DP), which provides rigorous guarantees against information leakage by introducing carefully calibrated randomness into the data or computation. While DP has been successfully applied in centralized settings with a trusted data curator, *Local Differential Privacy* (LDP) has gained attention for its stronger trust model, wherein each individual perturbs their own data before sharing. This eliminates the need for a central trusted party.

Applying LDP to graph learning introduces new challenges. Unlike standard tabular data, graph data is inherently relational—meaning that an individual's data may be intertwined with their neighbors' data. Existing research has addressed local privacy for node features and labels; however, comparatively little focus has been given to protecting the structural component of graphs—namely, the edge information—under local privacy constraints. Preserving privacy in such settings requires new approaches that can perturb graph structure while retaining utility for downstream tasks.

1.2 Problem Description

We consider an undirected graph $G = (V, E, X, Y)$, where the node set $V = V_L \cup V_U$ consists of a subset of labeled nodes V_L and unlabeled nodes V_U. The set E denotes the edges between nodes. The feature matrix $X \in \mathbb{R}^{|V| \times k}$ contains a k-dimensional feature vector for each node as its row. The label matrix $Y \in \mathbb{R}^{|V| \times c}$ has one-hot encoded vectors for labeled nodes, indicating their class membership among c possible classes, while rows corresponding to unlabeled nodes are set to zero vectors.

In our setting, each node has access to accurate information only about its own features, degree, and incident edges. However, it communicates a perturbed version of this data to an untrusted server. The

central challenge is to develop a noise injection strategy that privatizes the graph structure while still allowing the server to train a Graph Neural Network (GNN) that can accurately predict the labels Y of the unlabeled nodes. This requires carefully balancing privacy guarantees with the preservation of utility for downstream learning tasks.

1.3 Contributions

This thesis investigates the problem of training GNNs under local differential privacy constraints, with a specific focus on protecting graph structure. While prior works have considered feature and label privacy, structural privacy—especially under local assumptions—remains underexplored. Our key contributions are as follows:

- We propose two structure privatization strategies that locally perturb the graph's edges while preserving essential structural and semantic properties relevant to GNN learning.
- Our methods operate in a fully local setting, where each node independently privatizes its connections based only on limited local views and feature-based similarity.
- The proposed mechanisms maintain key structural characteristics, such as node degree, to ensure that the utility of the perturbed graph remains suitable for downstream tasks like node classification.
- To support comprehensive privacy protection, we integrate these structural mechanisms with existing approaches for feature and label privatization, thereby offering a unified framework for full node-level privacy.
- We conduct extensive empirical evaluation on multiple real-world datasets using various GNN architectures, and systematically analyze the impact of different privacy budgets and hyperparameters. Our results demonstrate that the proposed mechanisms strike an effective balance between privacy preservation and predictive performance.

Chapter 2

Preliminaries

2.1 Graph Neural Networks

GNNs are a powerful family of neural architectures designed to operate on graph-structured data, where information is inherently relational and non-Euclidean. At the core of GNNs lies a message-passing paradigm, wherein each node iteratively updates its representation by aggregating feature information from its neighbors.

Formally, at each layer l, a node $v \in V$ receives messages from its neighboring nodes $N(v)$, aggregates them using a permutation-invariant function such as mean or sum, and applies a non-linear update to compute its new embedding. This process can be abstracted as:

$$m_v^{(l)} = \text{Aggregate}^{(l)}\left(\left\{h_u^{(l-1)} : u \in N(v)\right\}\right), \tag{2.1}$$

$$h_v^{(l)} = \text{Update}^{(l)}\left(h_v^{(l-1)}, m_v^{(l)}\right), \tag{2.2}$$

where $h_v^{(0)} = x_v$ is the initial input feature vector of node v. The aggregation function must be invariant to the order of neighbors to preserve the graph's structural properties, and the update function typically consists of a linear transformation followed by a non-linearity such as ReLU.

After stacking L such layers, each node's final embedding $h_v^{(L)}$ incorporates information from its L-hop neighborhood. These embeddings can then be utilized for downstream tasks such as node classification, link prediction, or graph classification.

In the node classification setting, the final embedding of each node is passed through a task-specific decoder, often a simple linear layer followed by a softmax, to produce label predictions. The model is trained using a supervised loss, typically the cross-entropy loss computed over the labeled nodes, and optimized end-to-end using gradient-based methods.

Graph Convolutional Networks (GCNs), introduced by Kipf and Welling [1], represent one of the earliest and most widely adopted instances of the GNN framework. GCN simplifies the message-passing procedure by employing a linear neighborhood averaging scheme. The update rule for a single GCN layer is given by:

$$h_v^{(l)} = \sigma\left(\sum_{u \in N(v) \cup \{v\}} \frac{1}{\sqrt{\deg(v)\deg(u)}} W^{(l)} h_u^{(l-1)}\right), \tag{2.3}$$

where σ is an activation function such as ReLU, and $W^{(l)}$ is the trainable weight matrix at layer l.

In matrix form, the update rule for all nodes can be written as:

$$H^{(l)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l-1)}W^{(l)}\right), \tag{2.4}$$

where $\tilde{A} = A + I$ is the adjacency matrix with added self-loops and $\tilde{D}$ is the corresponding degree matrix. This formulation ensures that each node aggregates messages from itself and its immediate neighbors in a

normalized manner, preserving numerical stability and enabling effective learning.

The model is trained end-to-end using supervised loss over labeled nodes, and predictions for unlabeled nodes are made using the same decoder applied to their final embeddings.

2.2 Differential Privacy

As data continues to play a pivotal role in decision-making processes, preserving individual privacy during statistical analysis and machine learning has emerged as a significant challenge. Conventional anonymization techniques, such as removing personally identifiable information, often fail to provide robust privacy guarantees, particularly in the presence of adversaries equipped with auxiliary background knowledge.

DP addresses these shortcomings by ensuring that the output of any computation is statistically indistinguishable regardless of whether an individual's data is included in the input dataset [2]. This property implies that even an adversary with access to the entire dataset except one individual's record cannot confidently infer the inclusion or exclusion of that individual. Unlike earlier heuristics, DP offers a *worst-case* privacy guarantee without assuming limitations on the adversary's knowledge, making it especially suitable for applications involving sensitive information, such as healthcare, finance, and large-scale user analytics.

Definition 2.1 (ε-DP)**.** Let $\mathcal{M}$ be a randomized mechanism operating on datasets, and let D and D' be adjacent datasets differing in exactly one individual's data. The mechanism $\mathcal{M}$ satisfies ε-differential privacy if, for every possible output subset $S \subseteq \text{Range}(\mathcal{M})$, the following inequality holds:

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(D') \in S]$$

where the probability is taken over the internal randomness of $\mathcal{M}$.

While the standard (central) model of differential privacy assumes the presence of a trusted curator who collects and perturbs raw data before releasing results, such a trust assumption may not always be realistic. In scenarios where a central aggregator cannot be trusted, *Local Differential Privacy (LDP)* offers a compelling alternative. Under LDP, each individual perturbs their data locally—prior to transmission—using a randomized mechanism, ensuring that privacy is preserved at the source itself.

Definition 2.2 (ε-LDP)**.** A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ satisfies ε-local differential privacy if, for any pair of inputs $x, x' \in \mathcal{D}$ and for every subset $S \subseteq \mathcal{R}$, the following holds:

$$\Pr[\mathcal{M}(x) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(x') \in S]$$

This condition ensures that the mechanism's output distribution is nearly indistinguishable whether the input was x or x', thereby safeguarding individual-level privacy even before any data aggregation occurs.

A central concept in both DP and LDP is the *privacy budget* ε, which characterizes the trade-off between privacy and utility. A smaller ε corresponds to stronger privacy, as it forces output distributions for neighboring inputs to be more similar. In the extreme case where $\varepsilon = 0$, the mechanism is fully private and produces outputs independent of the input. Conversely, as $\varepsilon \to \infty$, privacy protection vanishes. Achieving lower ε generally necessitates injecting more noise, which can reduce the utility of the output. The optimal choice of ε is application-dependent, influenced by the sensitivity of the data and the tolerance for error. In practice, values of ε are often chosen from the interval $(0, 10]$, aiming to balance privacy protection and practical utility [3, 4].

Importantly, LDP imposes a stricter privacy constraint than central DP, as it guarantees privacy without assuming a trusted aggregator. Each user independently applies perturbation before sharing data, making LDP particularly valuable in decentralized or privacy-sensitive settings.

2.3 Edge-Level Local Differential Privacy in Graphs

DP has been extended to graph-structured data by defining appropriate notions of adjacency. In the centralized setting, two graphs G and G' are said to be *edge-adjacent* if they differ by exactly one edge. Alternatively, *node adjacency* refers to graphs that differ by a single node and all its incident edges. While centralized DP requires a trusted data curator to collect raw data and inject noise in a post-processing step, this assumption may not hold in decentralized environments, thereby motivating the use of LDP.

In the graph setting, *edge-level LDP* ensures that the presence or absence of a single edge in a user's neighborhood does not significantly influence the output of the randomized mechanism. A user's local neighborhood can be represented as a binary vector $\mathbf{b} = (b_1, b_2, \ldots, b_{|V|})$, where $b_i = 1$ indicates an edge between the user v and node v_i, and $b_i = 0$ otherwise.

Definition 2.3 (ε-edge LDP)**.** A randomized mechanism $\mathcal{M} : \{0,1\}^{|V|} \to \mathcal{R}$ satisfies ε-edge local differential privacy if for any two neighbor lists $\mathbf{b}, \mathbf{b}' \in \{0,1\}^{|V|}$ differing in exactly one bit, and for any possible output subsets $S \subseteq \mathcal{R}$, the following holds:

$$\Pr[\mathcal{M}(\mathbf{b}) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(\mathbf{b}') \in S].$$

A commonly used mechanism to achieve ε-edge LDP is the *Randomized Response* (RR) technique, originally introduced by Warner [5] and later adapted for graph data by Qin et al. [6]. In this scheme, each bit of the binary neighbor vector is independently flipped with probability $\frac{1}{e^{\varepsilon}+1}$, which ensures ε-edge LDP. However, such bitwise perturbation can lead to a substantial increase in the density of originally sparse graphs, thereby impairing the performance of downstream graph analysis and learning tasks.

To mitigate this issue, a set-based variant of edge-level adjacency has been proposed, where a node's neighborhood is treated as a set rather than a bit vector. In this formulation, two neighborhoods are considered adjacent if they differ in exactly one node. This approach offers a more structure-preserving alternative and is formalized as follows, adapted from Joshi et al. [7].

Let $b = \{v_1, \ldots, v_d\}$ and $b' = \{v'_1, \ldots, v'_d\}$ be two node sets of equal cardinality. These sets are considered adjacent if they differ in exactly one element; without loss of generality, suppose $v_1 \neq v'_1$, and $v_i = v'_i$ for all $i = 2, \ldots, d$.

Definition 2.4 (ε-edge set LDP)**.** Let $\varepsilon > 0$. A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ satisfies ε-edge set local differential privacy if, for all adjacent neighbor sets b, b', and for all possible output subsets $S \subseteq \mathcal{R}$, it holds that:

$$\Pr[\mathcal{M}(b) \in S] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(b') \in S].$$

This set-based perturbation model can be equivalently realized using the bitwise Randomized Response mechanism by flipping two bits in the binary vector. Notably, this formulation inherently preserves the node degree, which can be desirable for various graph-based learning tasks where degree distributions play a critical role.

While the definition of ϵ-edge set LDP provides a strong guarantee by bounding the influence of any single edge in the neighbor set, it can be overly strict in practical scenarios—especially when dealing with high-dimensional or sparse graphs. To allow for greater flexibility and to accommodate rare but significant deviations from the strict privacy bound, we introduce a relaxed variant known as (ϵ, δ)-edge

set LDP. This formulation tolerates a small probability δ under which the privacy guarantee may be violated, enabling improved utility while still preserving meaningful privacy protections.

Definition 2.5 ((ϵ, δ)-Edge Set LDP)**.** Let $\epsilon > 0$ and $\delta \in [0, 1)$. A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ satisfies (ϵ, δ)-edge set local differential privacy if, for all adjacent neighbor sets b, b' (i.e., sets differing in exactly one neighbor), and for all possible output subsets $S \subseteq \mathcal{R}$, it holds that:

$$\Pr[\mathcal{M}(b) \in S] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(b') \in S] + \delta.$$

Chapter 3

Privacy Mechanisms

In the local privacy framework discussed in problem statement, we propose a method for ensuring the privacy of node features, node labels, and graph edges, with a primary emphasis on **edge privacy**. As the edge perturbation mechanism relies on node feature information, the first step in our approach involves **privatizing the features** of all nodes in the graph.

3.1 Feature Privacy Approach

Ensuring feature privacy for each node is essential when exchanging feature vectors with neighboring nodes or transmitting them to an untrusted server. To achieve this, we apply the *Multi-Bit Mechanism*, consisting of the *Multi-Bit Encoder* and *Multi-Bit Rectifier*, as proposed by Sajadmanesh and Gatica-Perez (2021) [3].

In the Multi-Bit Encoder algorithm, each node v in the graph is assumed to have a k-dimensional feature vector $\mathbf{x}_v$, with each element lying within the range $[\alpha, \beta]$. The algorithm randomly selects certain dimensions and encodes their values as either -1 or 1, based on their closeness to the endpoints α and β, while the remaining dimensions are set to 0. The resulting encoded vector $\mathbf{x}_v^* \in \{-1, 0, 1\}^k$ serves as a biased estimator of the original vector. To remove this bias and recover an unbiased estimate of $\mathbf{x}_v$, the Multi-Bit Rectifier algorithm is subsequently applied.

Algorithm 1 Multi-Bit Encoder

Input: feature vector $\mathbf{x} \in [\alpha, \beta]^k$; privacy budget $\varepsilon > 0$; range parameters α, β; sampling parameter $m \in \{1, 2, \ldots, k\}$
Output: encoded vector $\mathbf{x}^* \in \{-1, 0, 1\}^k$
Let $\mathcal{S}$ be a set of m values drawn uniformly at random without replacement from $\{1, 2, \ldots, k\}$
for $i = 1$ **to** k **do**
    $s_i \leftarrow 1$ if $i \in \mathcal{S}$; otherwise $s_i \leftarrow 0$
    $t_i \sim \text{Bernoulli}\left(\frac{1}{e^{\varepsilon/m}+1} + \frac{x_i - \alpha}{\beta - \alpha} \cdot \frac{e^{\varepsilon/m}-1}{e^{\varepsilon/m}+1}\right)$
    $x_i^* \leftarrow s_i \cdot (2t_i - 1)$
end for
return $\mathbf{x}^* = [x_1^*, \ldots, x_k^*]^T$

Theorem 3.1.1. The Multi-Bit Encoder presented in Algorithm 1 satisfies ε-local differential privacy for each node.

Multi-Bit Rectifier. To obtain an unbiased estimate $\tilde{\mathbf{x}}$ from the encoded vector $\mathbf{x}^*$, the following rectification function is applied:

$$\tilde{\mathbf{x}} = \text{Rect}(\mathbf{x}^*) = \frac{k(\beta - \alpha)}{2m} \cdot \frac{e^{\varepsilon/m} + 1}{e^{\varepsilon/m} - 1} \cdot \mathbf{x}^* + \frac{\alpha + \beta}{2} \tag{3.1}$$

3.2 Edge Privacy (Approach 1)

To enforce edge-level local differential privacy, we propose a perturbation mechanism wherein each node locally alters its neighborhood by substituting a randomly selected subset of its immediate neighbors with structurally plausible alternatives. These substitute nodes are sampled from the node's 2-hop neighborhood and are selected based on a predefined similarity metric. This ensures that the perturbed graph structure retains its utility for downstream graph learning tasks while simultaneously offering privacy protection.

Let v be a node in the graph G. To privatize its edges, a random subset of neighbors in $N(v)$ is replaced with nodes from its 2-hop neighborhood that exhibit feature similarity to the removed nodes.

During the first message propagation step, each node transmits its feature vector—privatized using the *Multi-Bit Mechanism*—to its immediate neighbors. As a result, node v receives a collection of perturbed feature vectors from its neighbors, denoted as $\{\tilde{x}_u : u \in N(v)\}$, where $\tilde{x}_u$ represents the noisy version of node u's feature vector.

For example, in Figure 3.1, node v's neighborhood is given by $N(v) = \{a, b, c, d\}$. After the initial message passing round, node v obtains the perturbed feature set $\{\tilde{x}_a, \tilde{x}_b, \tilde{x}_c, \tilde{x}_d\}$ from its neighbors.

In the second round of message propagation, each node transmits to its neighbors the previously received perturbed features, augmented with a small set of dummy feature vectors. These dummy vectors are generated by independently sampling each feature from a standard normal distribution, scaling them by the global feature-wise standard deviation, and shifting them by the global feature-wise mean. Formally, each dummy feature vector is drawn from the multivariate Gaussian distribution $\mathcal{N}(\mu, \sigma^2 I)$, where μ and σ are the empirical mean and standard deviation, respectively, computed over each feature dimension in the dataset. The resulting vectors are clipped element-wise to lie within the interval $[\alpha, \beta]$, defined by the minimum and maximum values observed across all features.

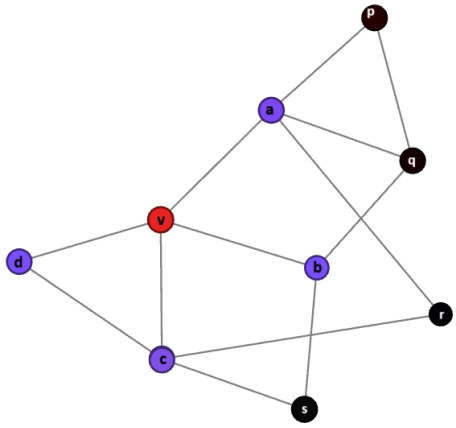

Figure 3.1: An undirected graph G with a vertex set $V = \{v, a, b, c, d, p, q, r, s\}$, and a corresponding set of feature vectors $X = \{x_v, x_a, x_b, x_c, x_d, x_p, x_q, x_r, x_s\}$, where each vector is of dimension k.

The number of dummy vectors appended by a node is a randomized function of its degree. Specifically, for a node with degree d, the base count is $\min(\lfloor d/2 \rfloor, f_{\max})$, where $f_{\max}$ is the maximum permitted number of dummy vectors. To obscure the degree information further, a random offset $\delta \in \{0, 1\}$ is added, yielding a final dummy count of

$$f = \min(\lfloor d/2 \rfloor + \delta, f_{\max}),$$

where δ is drawn uniformly at random. This randomized strategy prevents the exact degree of a node from being inferred based on the number of shared features.

In the example illustrated in Figure 3.1, during the second message propagation step, node v receives the following sets of perturbed feature vectors: from node a, the set $\{\tilde{x}_p, \tilde{x}_q, \tilde{x}_r, \tilde{x}_v\}$; from node b, $\{\tilde{x}_q, \tilde{x}_s, \tilde{x}_v\}$; from node c, $\{\tilde{x}_r, \tilde{x}_s, \tilde{x}_d, \tilde{x}_v\}$; and from node d, $\{\tilde{x}_c, \tilde{x}_v\}$. In addition, node v also receives dummy feature vectors from each of its neighbors a, b, c, and d, independently.

Following the message propagation phase, node v has access to the perturbed feature vectors of all nodes in its 2-hop neighborhood, along with the dummy vectors. More precisely, it holds the set

$$\{\tilde{x}_u : u \in N(v)\} \cup \{\tilde{x}_w : w \in N(u),\ \forall u \in N(v)\},$$

in addition to the dummy features received from its neighbors.

Node v then aggregates this collection, removes duplicate entries, and filters out any vectors corresponding to its immediate neighbors or to itself.

To privatize its edge set further, we apply the *Randomized Response* (RR) technique—a widely adopted method for achieving local differential privacy. The RR mechanism is applied to the neighbor list $N(v)$ of node v. Under a given privacy parameter ϵ, each bit in the neighbor list is flipped with probability

$$p = \frac{1}{1 + e^{\epsilon}}.$$

As defined in the preliminaries, the neighbor list $N(v)$ contains $\deg(v)$ entries with a value of 1, each representing a neighboring node of v.

Suppose the RR mechanism flips k such 1-entries to 0. Let the removed neighbors be $\{u_1, u_2, \ldots, u_k\}$. To replace these, node v searches for feature vectors within its previously collected (and filtered) set that are most similar to $\{\tilde{x}_{u_1}, \tilde{x}_{u_2}, \ldots, \tilde{x}_{u_k}\}$. The similarity is measured using either Euclidean distance or cosine similarity.

For each removed neighbor u_i, node v retrieves the top-k most similar feature vectors. To determine the true identity behind the most similar feature, node v sequentially queries its immediate neighbors, each maintaining a local mapping of features to node identities. If none of the neighbors can resolve the identity (implying the feature corresponds to a dummy), node v proceeds to the next most similar vector and repeats the query process. This continues until node v successfully identifies a node whose feature closely matches $\tilde{x}_{u_i}$. The identified node is then used to replace the removed neighbor u_i.

Returning to the illustrative example, assume that after applying RR, node v removes neighbors b and d. To restore the neighborhood, node v searches for the most similar feature vectors to $\tilde{x}_b$ and $\tilde{x}_d$ among the candidate set and replaces them with their corresponding identities.

This entire procedure ensures that every removed neighbor is substituted with a structurally consistent candidate, thereby finalizing the edge perturbation process for node v.

Algorithm 2 Edge Privacy Mechanism (Approach 1)

Input: Graph $G = (V, E)$ with node features X, privacy budgets ϵ_x, ϵ_e, maximum dummies $f_{\max}$, similarity metric sim $\in \{\text{cosine}, \ell_2\}$
Output: Perturbed graph G' with privatized edges
for each node $v \in V$ **do**
    $\tilde{x}_v \leftarrow \text{MBM}(x_v, \epsilon_x)$
end for
for each node $v \in V$ **do**
    Initialize local feature map: $\text{FeatureMap}_v \leftarrow \emptyset$
    for each neighbor $u \in N(v)$ **do**
        $\text{FeatureMap}_v[\tilde{x}_u] \leftarrow u$
    end for
end for
for each node $v \in V$ **do**
    Initialize $\text{ReceivedFeatures}_v \leftarrow \emptyset$
    for each neighbor $u \in N(v)$ **do**
        Receive features $\{\tilde{x}_w : w \in N(u)\}$
        Let $\alpha = \min(X)$ and $\beta = \max(X)$ across all node features
        $f_u \leftarrow \min(\lfloor \deg(u)/2 \rfloor + \delta, f_{\max})$, where $\delta \sim \text{Uniform}(\{0, 1\})$
        Generate f_u dummy features from $\mathcal{N}(\mu, \sigma^2 I)$ clipped to $[\alpha, \beta]$
        Add both real and dummy features to $\text{ReceivedFeatures}_v$
    end for
    Remove duplicates then Remove $\tilde{x}_v$ and $\{\tilde{x}_u : u \in N(v)\}$ from $\text{ReceivedFeatures}_v$
    Let $\text{CandidateFeatures}_v \leftarrow$ remaining features
end for
for each node $v \in V$ **do**
    Apply Randomized Response with ϵ_e on $N(v)$
    Let $R(v) \leftarrow \{u \in N(v) : \text{RR bit flipped}\}$
    for each removed neighbor $u_i \in R(v)$ **do**
        Compute similarity scores between $\tilde{x}_{u_i}$ and features in $\text{CandidateFeatures}_v$
        Let $\text{TopK}_v \leftarrow$ top-K features with highest cosine or lowest ℓ_2 score
        for each candidate feature $z_j \in \text{TopK}_v$ **do**
            for each neighbor $w \in N(v)$ **do**
                if $z_j \in \text{FeatureMap}_w$ **then**
                    $\text{identity} \leftarrow \text{FeatureMap}_w[z_j]$
                    if identity $\neq v$ and identity $\notin N(v)$ **then**
                        Replace edge (v, u_i) with $(v, \text{identity})$ in G
                        break
                    end if
                end if
            end for
            if edge replaced **then break**
            end if
        end for
    end for
end for
return G'

3.3 Edge Privacy (Approach 2)

In this variant of edge perturbation, we introduce a privacy-preserving mechanism that avoids explicit 2-hop message propagation and eliminates the need for dummy feature vectors. The protocol leverages feature similarity to identify suitable replacement nodes and utilizes randomized querying to achieve edge-level local differential privacy (LDP) while maintaining the utility of the graph.

As in Approach 1, each node begins by sharing its privatized feature vector with its immediate neighbors using the *Multi-Bit Mechanism*.

In the example graph shown in Figure 3.1, after the first round of message exchange, node v receives the perturbed features $\{\tilde{x}_a, \tilde{x}_b, \tilde{x}_c, \tilde{x}_d\}$ from its neighbors. Similarly, node a receives $\{\tilde{x}_p, \tilde{x}_q, \tilde{x}_r, \tilde{x}_v\}$, node b receives $\{\tilde{x}_q, \tilde{x}_s, \tilde{x}_v\}$, and so forth.

Each node v then applies the Randomized Response (RR) mechanism to its neighbor list $N(v)$, perturbing its adjacency by independently flipping each entry in the adjacency vector with probability $\frac{1}{1+e^{\epsilon}}$, where ϵ denotes the privacy budget. Let $R \subset N(v)$ denote the set of neighbors removed as a result of RR, and let $T = N(v) \setminus R$ represent the retained neighbors.

To obscure which neighbors were actually removed, node v sends the perturbed feature vectors corresponding to both R and T to its original neighbors in a randomly shuffled order. This randomized ordering ensures that no neighbor can infer with certainty whether it was included in or excluded from the perturbed neighborhood.

For instance, in the graph shown in Figure 3.1, suppose nodes b and d are removed from node v's neighbor list, while nodes a and c are retained. Node v then shares the perturbed features from both sets—$\tilde{x}_b, \tilde{x}_d$ (removed) and $\tilde{x}_a, \tilde{x}_c$ (retained)—with all of its original neighbors. The feature vectors are transmitted in a uniformly random order, preventing any neighbor from identifying whether it was part of the retained or removed set.

Upon receiving these feature vectors from node v, each neighbor of node v processes the two query sets R and T in the order they were received. A neighbor u responds only if it has at least as many neighbors (excluding v) as the number of feature vectors in the query. If this condition is met, node u identifies the most similar features from the perturbed features of its own neighbors (excluding v), which were obtained during the initial round of message exchange. The similarity is measured using either cosine similarity or Euclidean (ℓ_2) distance. For each query feature, the closest match from the candidate pool is selected and returned to node v.

In the example from Figure 3.1, neighbors a, b, and c of node v respond to both query sets $\{\tilde{x}_b, \tilde{x}_d\}$ and $\{\tilde{x}_a, \tilde{x}_c\}$ by returning the most similar feature vector from their own local neighborhoods (excluding v). In contrast, node d, having only one neighbor other than v, does not respond, as it lacks sufficient neighbors to process the queries.

Once node v collects the similarity responses from its neighbors, it filters the subset of features corresponding to the removed neighbors R. To avoid creating self-loops or redundant edges, it discards any feature vectors associated with itself or with its original neighbors. From the remaining candidates, node v selects the best-matching feature vector for each removed neighbor based on the specified similarity metric.

Subsequently, node v performs identity resolution. For each selected candidate feature, it queries its original neighbors one by one to determine the identity of the node associated with that perturbed feature. Each neighbor maintains a local mapping from features to node identities and checks whether the queried feature matches any of the features it observed during the initial round. This querying continues until a valid identity is discovered. If a successful match is found, node v replaces the edge $(v, \text{removed node})$ with a new edge $(v, \text{resolved identity})$.

In this way, the neighborhood $N(v)$ of each node is perturbed locally, ensuring a controlled and privacy-preserving modification of the graph structure.

Theorem 3.3.1. Both the Edge Privacy approaches (Algorithms 2 and 3) satisfy (ϵ, δ)-edge set local differential privacy.

Proof. Let us consider that, given a privacy budget ϵ, each node flips each bit in its neighbor list with probability $\frac{1}{1+e^\epsilon}$, as suggested in Qin et al. [6]. So, for $b, b' \in \{0, 1\}$,

$$\Pr[b \to b'] = \begin{cases} \frac{1}{1+e^\epsilon} & \text{if } b \neq b' \\ \frac{e^\epsilon}{1+e^\epsilon} & \text{if } b = b' \end{cases}$$

Let v be a node in the graph and $u \in \mathcal{N}(v)$. Then for $u' \in V$ (the set of all vertices), if $u \neq u'$,

$$\Pr[u \to u'] = \Pr(1 \to 0 \text{ for } u) \times \Pr(0 \to 1 \text{ for } u') = \left(\frac{1}{1+e^\epsilon}\right)^2$$

Therefore,

$$\Pr[u \to u'] = \begin{cases} \left(\frac{1}{1+e^\epsilon}\right)^2 & \text{if } u \neq u' \\ 1 - \left(\frac{1}{1+e^\epsilon}\right)^2 & \text{if } u = u' \end{cases}$$

Now we proceed to prove the theorem. Let $b = \{v_1, v_2, \ldots, v_d\}$, $b' = \{v'_1, v'_2, \ldots, v'_d\}$ be two neighbor sets that differ only in one element; assume without loss of generality that $v_1 \neq v'_1$. Then, given any output $s = \{s_1, s_2, \ldots, s_d\}$ of the mechanism M,

$$\frac{\Pr[M(b) = s]}{\Pr[M(b') = s]} = \frac{\Pr[v_1 \to s_1] \Pr[v_2 \to s_2] \ldots \Pr[v_d \to s_d]}{\Pr[v'_1 \to s_1] \Pr[v'_2 \to s_2] \ldots \Pr[v'_d \to s_d]} = \frac{\Pr[v_1 \to s_1]}{\Pr[v'_1 \to s_1]}$$

Previously we assumed that $v_1 \neq v'_1$. Now the following cases can occur:

—

Case I: s_1 is neither v_1 nor v'_1, i.e., $s_1 \neq v_1 \neq v'_1$

$$\frac{\Pr[v_1 \to s_1]}{\Pr[v'_1 \to s_1]} = \frac{\left(\frac{1}{1+e^\epsilon}\right)^2}{\left(\frac{1}{1+e^\epsilon}\right)^2} = 1 < e^\epsilon, \quad \forall \epsilon > 0$$

—

Case II: $s_1 = v_1$

$$\frac{\Pr[v_1 \to s_1]}{\Pr[v'_1 \to s_1]} = \frac{1 - \left(\frac{1}{1+e^\epsilon}\right)^2}{\left(\frac{1}{1+e^\epsilon}\right)^2} = (1+e^\epsilon)^2 - 1 = 2e^\epsilon + e^{2\epsilon}$$

According to the definition, the randomized mechanism M is said to satisfy (ϵ, δ)-edge set LDP if the following holds:

$$\Pr[M(b) = s] \leq e^\epsilon \Pr[M(b') = s] + \delta$$

That is, in this case:

$$\Pr[v_1 \to s_1] \leq e^\epsilon \Pr[v'_1 \to s_1] + \delta$$

which implies:

$$\frac{\Pr[v_1 \to s_1]}{\Pr[v_1' \to s_1]} \leq e^{\epsilon} + \frac{\delta}{\Pr[v_1' \to s_1]} \tag{3.1}$$

Now to satisfy Equation (3.1):

$$2e^{\epsilon} + e^{2\epsilon} \leq e^{\epsilon} + \frac{\delta}{\Pr[v_1' \to s_1]}$$

$$\Rightarrow e^{\epsilon} + e^{2\epsilon} \leq \frac{\delta}{\left(\frac{1}{1+e^{\epsilon}}\right)^2}$$

$$\Rightarrow e^{\epsilon}(1 + e^{\epsilon}) \leq \delta(1 + e^{\epsilon})^2$$

$$\Rightarrow \delta \geq \frac{e^{\epsilon}}{1 + e^{\epsilon}}$$

So, our choice for δ is:

$$\delta = \frac{e^{\epsilon}}{1 + e^{\epsilon}}$$

—

Case III: $s_1 = v_1'$

$$\frac{\Pr[v_1 \to s_1]}{\Pr[v_1' \to s_1]} = \frac{\left(\frac{1}{1+e^{\epsilon}}\right)^2}{1 - \left(\frac{1}{1+e^{\epsilon}}\right)^2} = \frac{1}{(1 + e^{\epsilon})^2 - 1} = \frac{1}{2e^{\epsilon} + e^{2\epsilon}} = \frac{1}{e^{\epsilon}(1 + e^{\epsilon})} < e^{\epsilon}, \quad \forall \epsilon > 0$$

—

Therefore, for Case I and Case III, the mechanism M satisfies ϵ-edge set LDP, and for Case II, it satisfies (ϵ, δ)-edge set LDP with $\delta = \frac{e^{\epsilon}}{1+e^{\epsilon}}$. Combining all three cases, we conclude that both of our approaches satisfy (ϵ, δ)-edge set LDP for $\delta = \frac{e^{\epsilon}}{1+e^{\epsilon}}$. □

3.4 Label Privacy

As the primary focus of our work is on edge privatization, we do not propose new mechanisms for label privacy. Instead, we adopt the label privatization strategy from the Drop algorithm introduced by Sajadmanesh and Gatica-Perez (2021) [3], which enables locally differentially private GNN training with perturbed labels using randomized response. For our implementation, we directly utilize the proposed label perturbation technique from their work.

Algorithm 3 Edge Privacy Mechanism (Approach 2)

Input: Graph $G = (V, E)$ with features X, budgets ϵ_x, ϵ_e, similarity metric sim $\in \{\text{cosine}, \ell_2\}$
Output: Perturbed graph G'

for each node $v \in V$ **do**
    Initialize $N(v)$ as neighbors of v
    Apply MBM to perturb x_v using ϵ_x
    Receive privatized features $\{\tilde{x}_u \mid u \in N(v)\}$
end for

for each node $v \in V$ **do**
    Initialize FeatureMap$_v \leftarrow \emptyset$
    for each $u \in N(v)$ **do**
        FeatureMap$_v[\tilde{x}_u] \leftarrow u$
    end for
end for

for each node $v \in V$ **do**
    Apply RR with ϵ_e to $N(v)$
    Let $R(v)$ be removed neighbors, $T(v) \leftarrow N(v) \setminus R(v)$
    Sample send_retained_first $\sim$ Bernoulli(0.5)
    if send_retained_first **then**
        first_set $\leftarrow \{\tilde{x}_u \mid u \in T(v)\}$, second_set $\leftarrow \{\tilde{x}_u \mid u \in R(v)\}$
        removed_position $\leftarrow$ "second"
    else
        first_set $\leftarrow \{\tilde{x}_u \mid u \in R(v)\}$, second_set $\leftarrow \{\tilde{x}_u \mid u \in T(v)\}$
        removed_position $\leftarrow$ "first"
    end if
    for each $u \in N(v)$ **do**
        Send both first_set and second_set to u
        u responds with most similar features from $N(u) \setminus \{v\}$
    end for
    Collect replies first_responses and second_responses
    if removed_position = "first" **then**
        candidate_features $\leftarrow$ first_responses
    else
        candidate_features $\leftarrow$ second_responses
    end if
    Remove duplicates, x_v, and $\tilde{x}_u$ for $u \in N(v)$ from candidate_features
    for each $u_i \in R(v)$ **do**
        Find most similar $z^* \in$ candidate_features to $\tilde{x}_{u_i}$ using sim
        for each $w \in N(v)$ **do**
            Query FeatureMap$_w[z^*]$ for identity
            if id is valid and not in $N(v) \cup \{v\}$ **then**
                Replace edge (v, u_i) with (v, id) in G
                break
            end if
        end for
    end for
end for

return Perturbed graph G'

Chapter 4

Experiments and Evaluation

We conduct comprehensive experiments to evaluate the privacy-utility trade-off of the proposed method for the node classification task. The evaluation spans a wide range of parameter settings, including similarity measures, feature privacy budgets, edge privacy budgets, and label privacy budgets. To thoroughly assess the effectiveness of our approach under diverse scenarios, we use a variety of GNN architectures encompassing traditional convolutional GNNs, graph attention networks. Specifically, we experiment with GCN(Kipf & Welling, 2017) [1], GraphSAGE(Hamilton et al., 2017) [8], GAT(Veličković et al., 2018) [9].

4.1 Datasets and Setup

These models are evaluated on widely-used benchmark datasets for node classification, including citation networks (Cora, Pubmed) [10] and social networks (LastFM [11]).

Cora and **Pubmed**: These are widely-used citation network datasets in the graph learning literature. In both datasets, each node represents a scientific publication, and an undirected edge between two nodes indicates a citation relationship between the corresponding papers. Node features are represented as bag-of-words vectors derived from the document contents. The class label of each node corresponds to the research topic or subject area of the paper. These datasets are primarily used for node classification tasks.

LastFM: This is a social network dataset constructed from user interactions on the LastFM music streaming platform. Nodes represent users from Asian countries, and edges denote mutual friendship relations among them. Each node is associated with a multi-hot feature vector indicating the musical artists liked by the user. The classification task involves predicting the home country of a user based on their listening preferences. Since the original dataset exhibits a high class imbalance, we restrict our evaluation to the top 10 most frequent classes to ensure a balanced and meaningful classification setup.

Dataset	Classes	Nodes	Edges	Features	Avg. Degree
Cora	7	2,708	5,278	1,433	3.90
Pubmed	3	19,717	44,324	500	4.50
LastFM	10	7,083	25,814	7,842	7.29

Table 4.1: Summary of graph datasets used in our experiments

All experiments are conducted using a 50:25:25 split for training, validation, and test sets, following the protocol of Sajadmanesh & Gatica-Perez (2021) [3]. All the GNN models have two graph convolution layers with a hidden dimension of size 16 and the SeLU activation function followed by dropout, and the GAT model uses four attention heads. Each model is trained for 100 epochs with a learning rate of 10^{-2}, weight decay of 10^{-3}, and a dropout rate of 0.5.

Experiments are performed using two similarity measures for neighbor selection: cosine similarity and L2 distance. These measures guide the replacement of neighbors during edge perturbation in the proposed methods.

We evaluate performance across various privacy settings by varying the edge privacy budget $\epsilon_e \in \{0.01, 0.1, 1, 2, 8\}$, the feature privacy budget $\epsilon_x \in \{0.01, 0.1, 1, 2, 3, \infty\}$, and the label privacy budget $\epsilon_y \in \{0.5, 1, 2, 3, \infty\}$. For each combination of ϵ_e, ϵ_x, and ϵ_y, we perform 4 independent runs using different random seeds and report the average results.

The KProp hyperparameters in the Drop mechanism are set according to the best configurations reported in Sajadmanesh & Gatica-Perez (2021) [3]: $K_x = 16, K_y = 2$ for Cora, and $K_x = 16, K_y = 0$ for both LastFM and PubMed.

4.2 Results and Analysis

We present the empirical results obtained from our proposed privacy mechanisms across three real-world datasets—Cora, Pubmed, and LastFM.

We analyze the performance of the proposed privacy-preserving mechanism under varying levels of feature, label, and edge privacy. In particular, we vary the feature privacy budget $\epsilon_x \in \{0.01, 0.1, 1, 2, 3, \infty\}$, the label privacy budget $\epsilon_y \in \{0.5, 1, 2, 3, \infty\}$, and the edge privacy budget $\epsilon_e \in \{0.01, 0.1, 1, 2, 8\}$. The settings $\epsilon_x = \infty$ and $\epsilon_y = \infty$ correspond to non-private baselines in which the respective LDP mechanisms—namely, the multi-bit encoder for features and the randomized response for labels—are disabled, and clean (unperturbed) data is used instead. These comparisons help isolate and understand the individual effects of each privacy mechanism on model performance.

We perform experiments using three popular GNN architectures: GCN, GAT, and GraphSAGE. Node classification accuracy is reported for each privacy configuration. The results, shown in Figure 4.1 & 4.2, highlight the trade-offs between privacy and utility across different combinations of privacy budgets and GNN models.

Across both figures corresponding to the L2 and cosine similarity mechanisms, it is evident that the test accuracy is significantly higher on the Pubmed dataset compared to Cora and LastFM. This suggests that model performance is dataset-dependent, and particularly influenced by the number of class labels. Notably, the Pubmed dataset, which contains only 3 classes, yields consistently higher accuracy than Cora and LastFM, which contain 7 and 10 classes, respectively.

For the Cora dataset, the results demonstrate a high degree of sensitivity to the feature privacy budget ϵ_x. When $\epsilon_x = \infty$, i.e., the raw features are used without any perturbation, the model achieves test accuracy of approximately 70-80%, even when the label and edge privacy budgets ($\epsilon_y \in \{2.0, 3.0\}$, $\epsilon_e \in \{0.01, 0.1, 1, 2, 8\}$) are set to strong privacy regimes. However, when the feature privacy budget is reduced to $\epsilon_x = 3.0$, the test accuracy drops significantly and fails to exceed 40% despite relatively relaxed privacy levels for labels and edges. These results highlight that the amount of noise introduced through feature perturbation plays a critical role in utility, and that a sufficiently high feature privacy budget ($\epsilon_x > 3.0$) is necessary to preserve classification performance on Cora. Increasing the edge privacy budget ϵ_e leads to a slight improvement in utility; however, its impact is notably less significant compared to that of the feature privacy budget ϵ_x, which plays a more dominant role in determining performance.

Pubmed emerges as the most robust dataset across different privacy regimes, consistently achieving relatively high accuracy (around 70–80%) even under moderate to strong privacy settings. Notably, when the feature privacy budget ϵ_x is not too small (i.e., $\epsilon_x > 1.0$), the accuracy reliably exceeds 60%.

Due to computational constraints, only a limited number of combinations were executed for the LastFM dataset. Nevertheless, based on the available results, the overall performance appears to be lower compared to Pubmed and Cora, with test accuracy typically remaining below 40%.

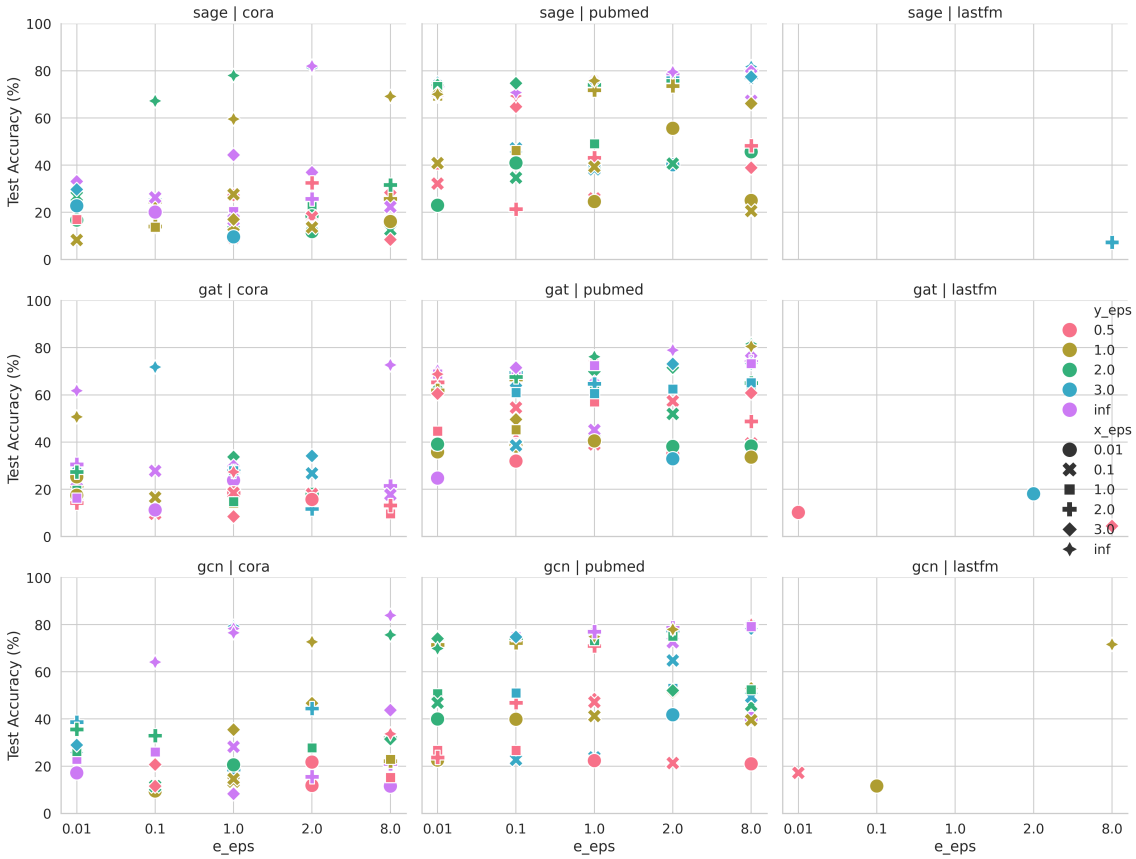

Figure 4.1: Grid plot showing test accuracy for different models under varying edge, feature, and label privacy budgets using L2 similarity.

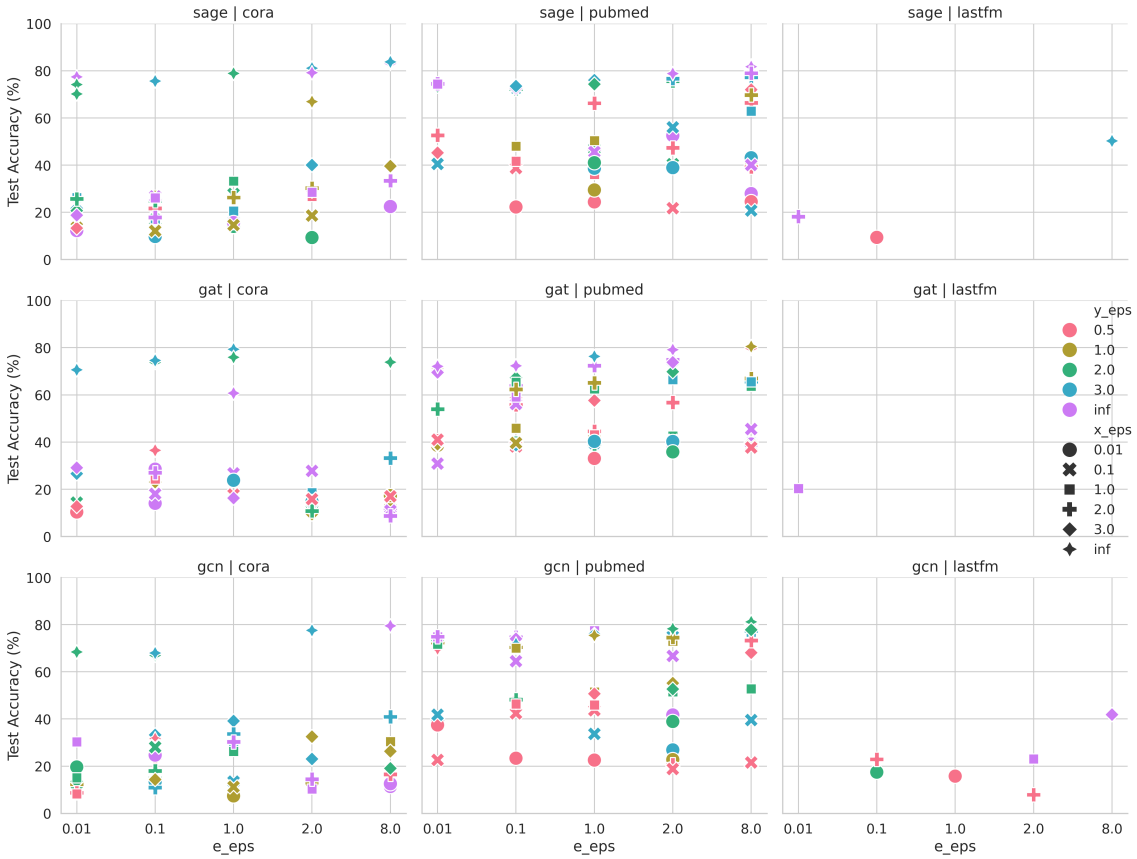

Figure 4.2: Grid plot showing test accuracy for different models under varying edge, feature, and label privacy budgets using cosine similarity.

Chapter 5

Conclusion and Future Work

In this work, I investigated the impact of LDP in the context of GNNs, where each node independently perturbs its own data without relying on a trusted server. Focusing on edge-level privacy, we proposed two novel mechanisms for structure privatization that preserve each node's degree while perturbing its neighborhood in a controlled and semantically meaningful manner. These approaches replace a subset of neighbors using similarity-based criteria, improving utility over purely random replacements. Our methods complement feature and label privatization strategies and collectively support a fully locally private learning pipeline. Empirical evaluations across multiple GNN models and real-world datasets demonstrate favorable privacy-utility trade-offs under various settings.

While the proposed methods provide locally differentially private edge perturbation mechanisms, there remain important challenges and open directions. A key issue arises during the identity resolution phase, where a node queries its neighbors to identify which node corresponds to a given feature vector. The current protocol assumes that neighbors truthfully respond if the identity exists in their local map. However, such a response reveals partial structural information about the neighbor's adjacency list, which conflicts with the goals of local privacy. Developing privacy-preserving identity resolution mechanisms that eliminate this disclosure remains a critical direction for future research.

Another promising avenue is the incorporation of personalized privacy preferences. In real-world networks, different users may have varying privacy requirements. Introducing a notion of mutual trust between nodes could enable selectively reduced noise when trusted neighbors interact, potentially enhancing utility without violating overall privacy guarantees.

Finally, robustness against adversarial or malicious nodes is a crucial consideration. In the current framework, all nodes are assumed to behave honestly during query and perturbation steps. Extending the framework to detect and mitigate privacy risks posed by adversarial nodes will be necessary for deploying such mechanisms in practical, open graph environments.

Bibliography

[1] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *CoRR* abs/1609.02907 (2016). arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907.

[2] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". In: *Foundations and Trends in Theoretical Computer Science* 9.3–4 (2014), pp. 211–407. DOI: 10.1561/0400000042.

[3] Sina Sajadmanesh and Daniel Gatica-Perez. "Locally Private Graph Neural Networks". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. New York, NY, USA: ACM, 2021, pp. 2130–2145. ISBN: 978-1-4503-8454-4. DOI: 10.1145/3460120.3485385.

[4] Fan Wu et al. "Linkteller: Recovering Private Edges from Graph Neural Networks via Influence Analysis". In: *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 2005–2024. DOI: 10.1109/SP46214.2022.9833642.

[5] Stanley L. Warner. "Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias". In: *Journal of the American Statistical Association* 60.309 (1965), pp. 63–69. DOI: 10.1080/01621459.1965.10480775.

[6] Zhan Qin et al. "Generating Synthetic Decentralized Social Graphs with Local Differential Privacy". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. New York, NY, USA: ACM, 2017, pp. 425–438. ISBN: 978-1-4503-4946-8. DOI: 10.1145/3133956.3134094.

[7] Rucha Bhalchandra Joshi, Patrick Indri, and Subhankar Mishra. "GraphPrivatizer: Improved Structural Differential Privacy for Graph Neural Networks". In: *Transactions on Machine Learning Research* (2024). ISSN: 2835-8856. URL: https://openreview.net/forum?id=lcPtUhoGYc.

[8] William L. Hamilton, Rex Ying, and Jure Leskovec. *Inductive Representation Learning on Large Graphs*. 2018. arXiv: 1706.02216 [cs.SI]. URL: https://arxiv.org/abs/1706.02216.

[9] Petar Veličković et al. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML]. URL: https://arxiv.org/abs/1710.10903.

[10] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. "Revisiting Semi-Supervised Learning with Graph Embeddings". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 40–48. URL: https://proceedings.mlr.press/v48/yanga16.html.

[11] Benedek Rozemberczki and Rik Sarkar. *Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models*. 2020. arXiv: 2005.07959 [cs.LG]. URL: https://arxiv.org/abs/2005.07959.

Chapter A

Deferred Theoretical Argument

A.1 Theorem 3.1

Theorem A.1.1. The Multi-Bit Encoder presented in Algorithm 1 satisfies ϵ-local differential privacy for each node.

Proof. Let $\mathcal{M}(x)$ denote the multi-bit encoder (Algorithm 1) applied on the input vector x. Let $x^* = \mathcal{M}(x)$ be the encoded vector corresponding to x. We need to show that for any two input features x_1 and x_2, we have:

$$\frac{\Pr[\mathcal{M}(x_1) = x^*]}{\Pr[\mathcal{M}(x_2) = x^*]} \leq e^{\epsilon}.$$

According to Algorithm 1, for any dimension $i \in \{1, 2, \ldots, d\}$, it can be easily seen that $x_i^* \in \{-1, 0, 1\}$. The case $x_i^* = 0$ occurs when $i \notin S$ with probability $1 - \frac{m}{d}$, therefore:

$$\frac{\Pr[\mathcal{M}(x_1)_i = 0]}{\Pr[\mathcal{M}(x_2)_i = 0]} = \frac{1 - \frac{m}{d}}{1 - \frac{m}{d}} = 1 \leq e^{\epsilon}, \quad \forall \epsilon > 0$$

In the case where $x_i^* \in \{-1, 1\}$, the probability of getting $x_i^* = 1$ ranges from $\frac{m}{d} \cdot \frac{1}{e^{\epsilon/m}+1}$ to $\frac{m}{d} \cdot \frac{e^{\epsilon/m}}{e^{\epsilon/m}+1}$ depending on x_i. Similarly, the probability of $x_i^* = -1$ also varies within the same range. Therefore:

$$\begin{aligned}\frac{\Pr[\mathcal{M}(x_1)_i \in \{-1, 1\}]}{\Pr[\mathcal{M}(x_2)_i \in \{-1, 1\}]} &\leq \frac{\max \Pr[\mathcal{M}(x_1)_i \in \{-1, 1\}]}{\min \Pr[\mathcal{M}(x_2)_i \in \{-1, 1\}]} \\ &\leq \frac{\frac{m}{d} \cdot \frac{e^{\epsilon/m}}{e^{\epsilon/m}+1}}{\frac{m}{d} \cdot \frac{1}{e^{\epsilon/m}+1}} = e^{\epsilon/m}\end{aligned}$$

Consequently, we have:

$$\begin{aligned}\frac{\Pr[\mathcal{M}(x_1) = x^*]}{\Pr[\mathcal{M}(x_2) = x^*]} &= \prod_{i=1}^{d} \frac{\Pr[\mathcal{M}(x_1)_i = x_i^*]}{\Pr[\mathcal{M}(x_2)_i = x_i^*]} \\ &= \prod_{i:x_i^*=0} \frac{\Pr[\mathcal{M}(x_1)_i = 0]}{\Pr[\mathcal{M}(x_2)_i = 0]} \cdot \prod_{i:x_i^* \in \{-1,1\}} \frac{\Pr[\mathcal{M}(x_1)_i \in \{-1, 1\}]}{\Pr[\mathcal{M}(x_2)_i \in \{-1, 1\}]} \\ &= \prod_{x_i^* \in \{-1,1\}} \frac{\Pr[\mathcal{M}(x_1)_i \in \{-1, 1\}]}{\Pr[\mathcal{M}(x_2)_i \in \{-1, 1\}]} \\ &\leq \prod_{x_i^* \in \{-1,1\}} e^{\epsilon/m} \\ &= e^{\epsilon}\end{aligned}$$

which concludes the proof. □