Maui: Black-Box Edge Privacy Attack on Graph Neural Networks

Haoyu He
GraphLab
The George Washington University
haoyuhe@gwu.edu

Isaiah J. King
GraphLab
The George Washington University
iking5@gwu.edu

H. Howie Huang
GraphLab
The George Washington University
howie@gwu.edu

ABSTRACT

Graphs are ubiquitous data structures with nodes representing objects and edges representing relationships between them. Graph Neural Networks (GNNs) have recently been proposed to study graph-structured data, but unfortunately, are susceptible to privacy leakage. This issue becomes more urgent as GNNs gain wide deployment in many real-world settings including social network analysis, bioinformatics, and cybersecurity. In this paper, we propose the first link inference attack that can compromise user data under the most difficult security settings, which we call Maui. We demonstrate that private edge information can be inferred by a malicious user with a black-box approach. Extensive experiments on six real-world datasets show our attacks conduct effective link inference attacks in various scopes. Our attack achieves significant performance improvements over the current state-of-the-art. When targeting 2-layer Graph Convolution Networks, for inferring edges of a single node, our attack outperforms the best existing method by 12.0%, increasing from 83.7% to 95.7%; when inferring edges of the entire graph, our attack achieves a 19.6% improvement, from 67.7% to 87.3%. Our results underscore the need for countermeasures against privacy attacks in GNNs, as they can reveal rich information about graph structures.

KEYWORDS

Graph Neural Networks, Link Inference Attack, Privacy

1 INTRODUCTION

Graphs are ubiquitous data structures for modeling relationships. Nodes represent discrete objects and edges represent connections between them. Practitioners use them to model data in various domains, such as social media graphs for recommendation [73], biological knowledge graphs for drug repurposing [24], and text graphs for relation extraction [45]. In recent years, researchers have proposed Graph Neural Networks (GNNs) to leverage graph-structured data with compelling results in many domains [64]. For example, Uber Eats implements GraphSAGE [20] on a user-item network to boost personalized recommendation; Google Maps deploys a GNN estimator on a road network to predict the expected time of arrival [10]; AWS applies Relational Graph Convolutional Networks [49] to a COVID-19 knowledge graph to accelerate drug research [58]. With so much real-world, and sometimes highly sensitive data being processed by these models, it is imperative to view them as a threat vector. How might an attacker exploit them to steal private data, and what measures can we take to stop them?

Table 1: Comparison of existing attacks whereas Maui requires no prior knowledge. We list the required knowledge for different attacks. The underlined knowledge indicates white-box access to the GNNs. (● = required; ○ = not required)

Attack	Feature/Label	Embedding	Spectral GNNs	Gradient
Decoder [12]	●	●	○	○
GraphMI [80]	●	●	●	●
LSA2 [21]	●	○	○	○
LinkTeller [62]	●	○	○	○
Maui	○	○	○	○

Traditional machine learning (ML) models are susceptible to attacks that reveal sensitive information [6, 36] and GNNs are no different. As GNNs are extensively deployed in real-world systems, privacy leakage has become a serious problem [8, 22, 60, 76, 79]. Privacy leakage of an attributed graph can be severe because, in addition to node and edge features, the graph structure itself is at risk. Even worse, because individual graph components possess rich information and are related to each other an attacker may infer the information of one component based on others [1, 18]. Additionally, with enough preliminary information, an attacker can conduct inference attacks to obtain sensitive and private information of graphs [21] and models [60].

In this paper, inspired by recent works [4, 21, 30, 37, 38, 51, 62], we consider the situation where graph data is *vertically partitioned* into two parts: graph structure (edges), and node features. These partitions are owned by different data holders and an ML platform is used for training and inference. Some more recent prior works have discussed security in the vertical partition paradigm for machine learning generally [46, 65, 68, 69], but as it is still an emerging field, fewer specifically consider the graph domain [4, 62]. With the growing popularity of GNN models, many inference servers are expanding to accommodate APIs for vertically partitioned graph data (Amazon Neptune[1], NVIDIA Triton[2], etc.). Once these services are implemented, the edge data is vulnerable to privacy leakage. Prior works have shown that with very little knowledge about the node features, classes, or the model, edge information can be stolen [21, 38, 62]. We will show that even with *no* prior knowledge, the model can still leak rich private information.

It is important to note that inference attacks against traditional machine learning models [16, 38] cannot be applied to graph data,

Proceedings on Privacy Enhancing Technologies 2024(4), 364–380

https://doi.org/10.56553/popets-2024-0121

[1] https://aws.amazon.com/neptune/
[2] https://developer.nvidia.com/nvidia-triton-inference-server

as they are designed for Euclidean data. Additionally, existing GNN-based link inference attacks fail to conduct effective attacks under more constrained settings. We aim to attack under the hardest possible settings. We assume that the GNN is a black-box, so the adversary has no way of knowing any of its internal parameters, or gradient information, only its output. This means prior works such as GraphMI [80], which makes the adjacency matrix a learnable parameter and relies on the backpropagation from GNNs, and Decoder [12], which relies on intermediate node embeddings, are unable to attack. In a more practical setting, let us assume that the adversary has only partial, or no node features or labels, e.g., users in a social network may not expose their personal information. The attacker only has API access to the ML platform. This makes approaches such as Link Stealing Attack-2 (LSA-2) [21], and LinkTeller [62] infeasible, as both rely on node features. This attack-space, "typically seen in the case of machine learning as a service" [12, p. 78], is what we refer to as the hardest setting. To further illustrate this, Table 1 shows the knowledge required for an adversary under several contemporary link inference attacks. It is also worth noting that Li et al. [33] has proposed a novel link inference attack without access to the node properties (including features and labels). However, this attack uses node injection approach to infer the connection between existing nodes, which involves creating new nodes and edges, and is unreasonable in our motivating scenario where the attackers cannot access the graph structure.

We also note that existing attacks fail to correctly identify edges for the whole graph. Works such as LinkTeller [62] which estimate gradients through node perturbations make the flawed assumption that all edges propagate the same amount of information, so they adopt a top-k strategy for predicting edges for the whole graph. This ignores the fact that nodes with more connections receive less information from each neighbor due to the aggregation scheme of GNNs [17, 66], making it difficult to detect edges in graphs with irregular density.

To address these problems, we propose MAUI, an attack that does not rely on any knowledge of node features or the computation during the prediction process; the attacker can only access the inference API as a black box and view its predictions. MAUI aims to infer whether an edge exists between two nodes. Specifically, MAUI first estimates the influence values locally for each node, then converts the local influence values into global influence values, so edges can be identified precisely for the whole graph. We study the privacy leakage of the graph structure from the inference stage of GNNs, when the graph is vertically partitioned into graph structure and node features.

Our experimental results show that MAUI significantly outperforms state-of-the-art attacks. When targeting 2-layer Graph Convolution Networks, for inferring edges of a single node (local attack), our attack outperforms the best existing method by 12.0%, increasing from 83.7% to 95.7%; when inferring edges of the entire graph (global attack), our attack achieves a 19.6% improvement, from 67.7% to 87.3%. The results further show that our attack can be deployed on various GNNs. Moreover, we demonstrate that MAUI is robust to model adaptations and defense mechanisms. Overall, we observe that GNN predictions can leak rich private link information even under the black-box setting, which suggests the necessity for corresponding defensive actions.

Table 2: Summary of notations.

Notation	Description
G	A graph with nodes and edges
V	Nodes in graph G
E	Edges in graph G
X	Node features of V
Y	Node labels of V
$d(\cdot,\cdot)$	Distance between nodes
$\Phi(\cdot,\cdot,\cdot)$	A GNN model
$f(\cdot,\cdot)$	An inference API derived from Φ
P	Prediction probabilities from Φ
H	Node representations from Φ
$\mathcal{I}$	Influence nodes based on Φ
Θ	Influence values of node pairs

In summary, we make the following contributions:

- **Novel privacy attack.** We propose the first link inference attack that operates under the most difficult security setting, using only the inference API and no node features or model information. This attack can effectively compromise edge information in graph-structured data.
- **Improved global attack.** We observe that existing attacks have limited success in conducting effective global attacks on the whole graph. To overcome this limitation, we introduce a normalization strategy that allows us to obtain globally compatible influence values.
- **Adaptation to constraints.** We design different strategies for MAUI to balance time complexity and attack performance. We provide three strategies in total and demonstrate that these strategies all outperform existing attacks.
- **Extensive experimental evaluation.** We conducted extensive experiments on six real-world datasets using commonly used GNN models. The experimental results demonstrate the effectiveness of MAUI from both local and global perspectives, significantly outperforming state-of-the-art attacks in precision, compatibility and robustness. We also show that existing defense mechanisms do not fully alleviate privacy concerns. Our work highlights the need for better privacy-preserving techniques for GNNs.

2 BACKGROUND

Notations. Let $G = (V, E)$ represent a graph with V denoting the nodes and E as the edge list. The element $e_{i,j}$ in the edge list represents the existence of an edge from node v_i to v_j. An attributed graph is a graph with associated attributes. In this paper, we consider nodes to be associated with node features X and labels Y.

We summarize the frequently used notations of this paper in Table 2. Capital letters represent a set or a matrix, and the corresponding lowercase represents a single element. Bold lowercase represents a vector. For example, $e_{i,j}$ indicates an edge from node v_i to v_j; $\boldsymbol{x}_i$ represents the node feature vector of v_i.

Graph Neural Networks. Graph Neural Networks (GNNs), Φ, are machine learning models working on graph-structured data [20,

29, 54]. They take an attributed graph as input and employ a neighborhood aggregation scheme through edges to learn node representations Z for different purposes. Neighborhood aggregation is conducted in each layer of a GNN model, which can be formulated into three essential steps. First, for each edge $e_{j,i}$, GNNs compute a message from the representations of $\{v_i, v_j\}$ in the previous layer: $\boldsymbol{m}_{j,i}^{l} = \textsc{Msg}(\boldsymbol{h}_i^{l-1}, \boldsymbol{h}_j^{l-1}, e_{j,i})$, where $\boldsymbol{h}_i^{l-1}$ is the representation of v_i in the $(l-1)$-th layer. Second, for each node v_i, GNNs aggregate all the messages from v_i's neighbor nodes $\mathcal{N}_i$: $\boldsymbol{m}_i^{l} = \textsc{Agg}(\{\boldsymbol{m}_{j,i}^{l} \mid v_j \in \mathcal{N}_i\})$. Finally, GNNs update the representation for each node v_i from the aggregated message and its previous representation: $\boldsymbol{h}_i^{l} = \textsc{Update}(\boldsymbol{m}_i^{l}, \boldsymbol{h}_i^{l-1})$. Specifically, the input representation $\boldsymbol{h}_i^{0}$ is the node feature $\boldsymbol{x}_i$, and the final representation $\boldsymbol{z}_i$ is the representation $\boldsymbol{h}_i^{L}$ from the last layer L.

The final representations are used for different tasks, such as node classification [29], graph classification [74], and link prediction [77]. Here, we focus on the node classification task. In general, a fully connected layer takes the final representations and computes the node predictions: $\boldsymbol{p}_i = \sigma(\text{FC}(\boldsymbol{z}_i))$, where σ is the activation function to normalize the input vector into probability distributions, e.g., Softmax. The prediction $\boldsymbol{p}_i$ is a vector containing the prediction probabilities of each class.

3 PROBLEM FORMULATION

This problem is based on the vertical partition of an attributed graph, where the graph structure and node features are distributed to at least two data holders. During the inference process, users with API access make requests for node predictions by sending node features to the inference API, which uses edge data held by other users to generate a probability vector representing the prediction probabilities for each class. Importantly, the inference API does not provide any information other than the corresponding predictions. We assume users have unlimited calls to the inference API, as it is reasonable to anticipate that either the model owner or the ML platform would grant complete API access within the semi-trusted user threat model. In this work, we wish to study how much of the graph structure can be inferred by a user who has only API access to the ML platform.

3.1 Motivating Example

Consider the scenario illustrated in Figure 1. Alice is a social media company that holds a great deal of proprietary relational information. She has two clients, Bob and Mallory, marketing companies that collect user preference data. No party wishes to share their data publicly, but they do wish to collaborate to generate better user profiles. To accomplish this, they use an ML platform to train a GNN on their vertically partitioned data. The GNN model takes as input Bob and Mallory's node features and using Alice's relational data, outputs predictions about those users' shopping habits, personality traits, etc..

Alice's clients follow the semi-trusted user threat model; we expect each entity to be "honest, but curious". They always securely store data and follow the agreed-upon protocol, but they may curiously infer from any information they receive [72]. Bob uses the API as expected, recalculating user profiles as his data is updated, but Mallory repeatedly makes new inference requests on the same nodes but sends slightly perturbed features each time. We will show how Mallory can reconstruct a very good approximation of Alice's proprietary edge data using only the API responses.

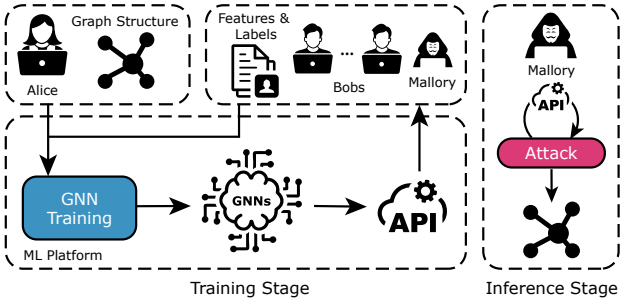


Figure 1: A motivating scenario: at the training stage, a GNN model is trained with edge data from Alice and node data from several clients, including Bob and Mallory, and deployed on an ML platform. At the inference stage, malicious users can infer the private graph structure via the inference API.

Given the way that inference as a service is currently implemented, it is likely that the scenario we outlined above exists in the wild. In fact, in the documentation for Amazon Neptune they show exactly how a customer with an API key can change node features and get new inference results from a pretrained model–all that is needed to implement our attack. Many marketing services follow this model of vertically partitioned data, and indeed, there is no shortage of such services on the AWS, Azure, or Google Cloud marketplaces, as well as in the literature [4, 7, 40].

3.2 Threat model

Attackers' goal. The attackers aim to infer whether an edge exists between a pair of nodes in the target graph during the inference stage.

Attackers' capability. This paper focuses on the situation where a holder of the inference API is an adversary. Since the holder only has partial or no knowledge of the node features, we assume the attackers possess only the identities of the nodes and the black-box access to the inference API of a GNN model. The attackers determine the connections of node pairs via consecutive API calls and prediction analysis. The attacker may also leverage external knowledge about the graph's topology and the characteristics of the nodes to improve the attack's accuracy. To summarize, the attackers' capability includes the knowledge of node identities, the black-box access to the inference API of GNNs, and the collections of corresponding prediction probabilities.

3.3 Maui

For simplicity, we formulate the target GNN model as $P = \Phi(V, E, X)$. The inference API is then defined as $f(V, X) = \Phi(V, E, X)$. Given a black-box access to the inference API, Maui aims to infer whether there exists an edge/link between a pair of nodes. Specifically, for a node pair (v_i, v_j), Maui provides an influence value $\theta_{i,j}$ indicating the influence of v_i on v_j from the inference API. For the whole

graph, we formulate MAUI as

$$\Theta = \textsc{Maui}(f, V). \tag{1}$$

With the estimated edge quantity, certain edges of the target graph can be inferred based on the influence values of the whole graph Θ.

4 DESIGN DETAILS OF MAUI

4.1 Attack Overview

Figure 2 illustrates the overall workflow of MAUI. Before the attacks, MAUI initializes nodes with the same random features for each node. In practice, it is possible for attackers to create realistic node features with knowledge such as node feature distributions and existing node features, but we do not assume such background knowledge. Then, MAUI takes the generated node features $\tilde{X}$, and perturbs them, generating $\hat{X}$. Next, it queries the inference API $f(\cdot,\cdot)$ with perturbed $\hat{X}$ on certain nodes. Finally, using the difference between the two API responses, it computes the influence values Θ for each node pair $\{(v_i, v_j)|v_i, v_j \in V\}$, which indicate the likelihood of a connection. With estimated edge density, MAUI connects node pairs based on the influence values in a global attack on the whole graph.

In this section, we will show how MAUI uses these influence values to infer all connections to a single node v_i. Then, we will build upon this, and show how MAUI can be used to estimate every edge in the graph.

4.2 Local Node Attack

If two nodes v_i and v_j are at least $L+1$ (including ∞) hops away, they have no influence on each other in an L-layer GNN model [62]. That is, a change to v_i's features $\boldsymbol{x}_i$ will not influence the model's prediction for v_j, $\boldsymbol{p}_j$. Thus, for an L-layer GNN model, we can find nodes within L hops by feature perturbation. Let the set of *influence nodes*, $\mathcal{I}_i$ denote nodes that are influenced by a perturbation to the features of v_i, not including v_i itself. To infer the connections of v_i, we only need to consider its influence nodes $\mathcal{I}_i$.

Intuitively, if two nodes are closer, the information of one node will be passed to the other through more paths. In addition, the information can reach the other node in an earlier layer, entailing a stronger signal. Because of the neighborhood aggregation scheme, less information from a node will be retained as it is passed to deeper layers. In summary, two nodes within smaller distances can propagate more information about themselves to each other, as well as produce more influence on each other's predictions. Therefore, we measure such influence as an indicator to uncover connections. If the influence value is bigger, it is more likely that two nodes are connected.

Based on this intuition, we can estimate the influence from v_j to v_i by measuring the change of $\boldsymbol{p}_i$ after setting $\boldsymbol{x}_j$ to zeros. Let $\hat{X}$ be a copy of $\tilde{X}$ except $\hat{\boldsymbol{x}}_j \leftarrow \mathbf{0}$. We calculate the influence value from v_j to v_i as:

$$\theta_{i,j} = \left\| f_i(V, \tilde{X}) - f_i(V, \hat{X}) \right\|_2, \tag{2}$$

where $f_i(\cdot,\cdot)$ represents the prediction of v_i and $\|\cdot\|_2$ is the L^2-norm. Different from LinkTeller, we set $\boldsymbol{x}_j$ to zeros rather than perturb it with small values for the concern of compatibility and robustness. For example, LinkTeller is not applicable to models with feature normalization. We refer to v_i as the *target node* and v_j as the *subject node*.

The impact of graph structure cannot be ignored during GNN computation. First, because nodes interact with each other through neighborhood aggregation, and the computation is not linear, changes in a target's prediction are not guaranteed. Second, the amount of information propagated is not necessarily proportional to the distance between nodes; node degree has a large effect as well. For example, a 2-hop densely connected subject node may propagate more information than a 1-hop trivially connected node simply because it has more edges sending messages.

To alleviate the first problem, we can estimate the influence value $\theta_{i,j}$, after removing the node features of v_i's influence nodes $\mathcal{I}_i$, except the subject node's features $\boldsymbol{x}_j$. By doing so, only the messages from the subject node can be passed to the target node. To account for the second problem, we present Proposition 1 to help improve the estimation for influence values in general. Our goal is to make sure the node pairs with smaller distances obtain larger influence values.

Proposition 1. *If two nodes v_i and v_j are l ($l < L$) hops away, given a GNN model of L layers, the intersection of their influence nodes $V^{(I)} = \mathcal{I}_i \cap \mathcal{I}_j$ covers their entire neighbor nodes within $L - l$ hops (excluding v_i and v_j).*

PROOF. We use $d(\cdot,\cdot)$ to denote the distance between two nodes. Let v_k be a node that is not v_i or v_j such that $d(v_i, v_k) \le L - l$ and $d(v_i, v_j) = l$, we will prove $v_k \in V^{(I)}$. From the given condition, because $v_k \in \mathcal{I}_i$, we must prove $v_k \in \mathcal{I}_j$. To prove $v_k \in \mathcal{I}_j$, we need to prove $d(v_j, v_k) \le L$.

The distance between v_j and v_k is measured from the shortest path between them. We consider two situations: 1) this path contains v_i; 2) this path does not contain v_i. In the first situation,

$$d(v_j, v_k) = d(v_j, v_i) + d(v_i, v_k) \le L.$$

In the second situation,

$$d(v_j, v_k) < d(v_j, v_i) + d(v_i, v_k) < L. \qquad \square$$

For example, suppose v_i and v_j are $l = 3$ hops away and the influence nodes are within $L = 5$ hops. Then the intersection $V^{(I)}$ covers all the 1- and 2-hop neighbors of v_i and v_j. If the two nodes were closer to each other, the intersection would contain a larger range of their neighbor nodes and other nodes with smaller distance to them.

To better distinguish the influence values of nodes at different distances from v_i, rather than removing the node features of all the influence nodes $\mathcal{I}_i$, we only set the node features $X^{(I)}$ of the intersection $V^{(I)}$ to zeros. Through the information propagation, if the subject node is closer to the target node, noise from a wider range of nodes will be mitigated. This noise mitigation is significant because for each hop the subject node is closer to the target, and the range of neighbor nodes in $V^{(I)}$ grows 1-hop larger. In a graph structure, the number of $(l+1)$-hop neighbors is usually exponentially larger than the number of l-hop neighbors.

The process for local attacks can be found in Algorithm 1. In summary, to attack a single target node v_i, we first iterate through each $v_j \in \mathcal{I}_i$, and calculate their intersection of influence nodes,

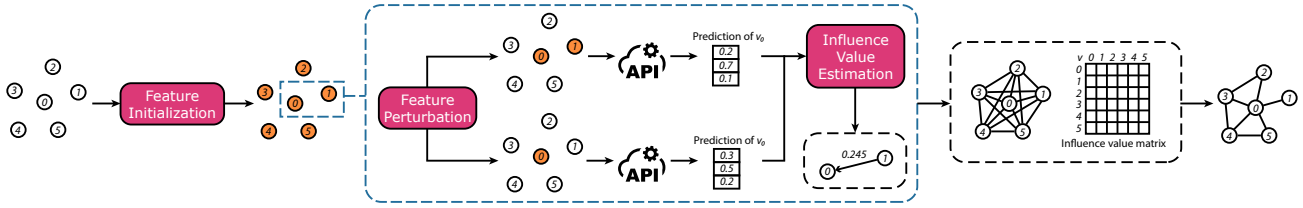


Figure 2: The workflow of Maui involves three steps. First, uniform features are randomly assigned to each node. Second, influence values between pairs of nodes are estimated using feature perturbation. Finally, edges are identified from the influence values by setting an estimated density and connecting the top pairs of nodes with the largest influence values.

Algorithm 1: Maui

Input: The inference API $f(\cdot,\cdot)$; the target nodes $V^{(T)} \subseteq V$
Output: The influence values Θ ($\tilde{\Theta}$)

1 Initialize random uniform node features $\tilde{X}$
2 Find all the influence nodes $\{\mathcal{I}_i | \forall v_i \in V\}$
3 $\Theta \leftarrow \{0\}^{|V^{(T)}| \times |V|}$
4 **foreach** $v_i \in V^{(T)}$ **do**
5 **foreach** $v_j \in \mathcal{I}_i$ **do**
6 $\hat{X} \leftarrow \tilde{X}$
7 $V^{(I)} \leftarrow \mathcal{I}_i \cap \mathcal{I}_j$
8 Set $\hat{X}^{(I)}$ to zeros
9 $\boldsymbol{p}_i \leftarrow f_i(V, \hat{X})$
10 Set $\hat{\boldsymbol{x}}_j$ to zeros
11 $\boldsymbol{p}'_i \leftarrow f_i(V, \hat{X})$
12 $\theta_{i,j} \leftarrow \left\|\boldsymbol{p}_i - \boldsymbol{p}'_i\right\|_2$
13 $\tilde{\boldsymbol{\theta}}_i = \boldsymbol{\theta}_i / \max \boldsymbol{\theta}_i$ ▷ global attack

$V^{(I)}$ (lines 5–7). Then, we set the features for each node in the intersection to zeros, leaving the rest as they were randomly initialized, and calculate the predictions $\boldsymbol{p}_i$ (lines 8–9). Next, we repeat the process, this time with $\boldsymbol{x}_j$ set to zeros as well to calculate $\boldsymbol{p}'_i$ (lines 10–11). Finally, we calculate the influence value of v_j on v_i, $\theta_{i,j}$ by taking the L^2-Squared distance between the different predictions (line 12). These values can be used on their own for this local attack, or as we will show, be further analyzed to estimate the graph in totality.

4.3 Global Graph Attack

To identify the edges for the whole graph, we cannot directly apply the top-k strategy or a classification threshold to the influence values Θ. Nodes with many edges receive relatively less information from their neighbors than those with fewer edges. This leads to smaller influence values. Another challenge is that during neighborhood aggregation, GNNs normalize the aggregated neighbor information for better performance. Furthermore, the estimation for influence values is based on normalized GNN predictions. As a result, we can only compare the influence values for a single target node. That is, only $\{\theta_{i,j} | v_j \in \mathcal{I}_i\}$ are compatible because they represent the influence towards v_i. Thus, for node pairs of completely different nodes, a larger influence value does not guarantee a higher probability of connection.

Proposition 2. *If $\boldsymbol{\theta}_i$ contains nonzero values, then v_i is connected by at least one edge.*

Proof. Assume v_i is not connected and there exists a node v_j, which is not connected to v_i and $\theta_{i,j} \neq 0$. Due to no connection, we have $d(v_i, v_j) = \infty$. Since $\theta_{i,j}$ is nonzero, according to Proposition 1, $d(v_i, v_j) \leq L$ which is a contradiction. Therefore, the assumption is false. □

Based on Proposition 2, we first find the member of $\mathcal{I}_i$ with the greatest influence on v_i. We assume the node pairs with the greatest influence values are connected. If the influence values of the remaining node pairs are close to the greatest influence value, these node pairs are more likely to be connected. Intuitively, we can measure such closeness and connect node pairs with the closest influence values to the corresponding strong influence values. To globally measure the closeness, we adopt min-max normalization to the influence values for each node:

$$\tilde{\boldsymbol{\theta}}_i = \frac{\boldsymbol{\theta}_i - \min \boldsymbol{\theta}_i}{\max \boldsymbol{\theta}_i - \min \boldsymbol{\theta}_i}. \tag{3}$$

In practice, $\min \boldsymbol{\theta}_i = 0$, so the equation is simplified to $\tilde{\boldsymbol{\theta}}_i = \boldsymbol{\theta}_i / \max \boldsymbol{\theta}_i$. If the influence value is closer to 1, it is closer to the corresponding strong influence value. Now we can globally compare the influence values and connect the top-k node pairs for the whole graph.

In real life, the attackers may estimate the edge density of a target graph, which helps them determine the number of edges $\hat{m}$ to recover. When a partial graph is accessible, the attackers can first find the influence values and then estimate the threshold or the edge density directly, according to different situations. The attackers can also guess the edge density from other accessible datasets that belong to similar categories. For example, attackers can use information from public social media accounts to approximate the density of a social network.

4.4 Different Estimation Strategies

Efficient strategy. Like LinkTeller, the time complexity can be reduced by computing the outgoing influence $\theta_{j,i}$ from node v_i, rather than the incoming influence ($\theta_{i,j}$). With the perturbation of one node, the outgoing influence can be estimated simultaneously. Using Equation (2), every outgoing influence value from v_i to $\mathcal{I}_i$ can be computed with only two queries of the inference API. Our goal is to find its outgoing influence values for a node v_i, $\{\theta_{j,i} | v_i \in \mathcal{I}_i\}$, without exhaustive traversal. However, as mentioned previously, using Equation (2) only is not accurate considering the nonlinearity

Algorithm 2: MAUI - Efficient

Input: The inference API $f(\cdot,\cdot)$; the nodes of a graph V; the window of sampling times α, β
Output: The influence values Θ ($\tilde{\Theta}$)

1 Initialize random uniform node features $\tilde{X}$
2 Find all the influence nodes $\{\mathcal{I}_i | \forall v_i \in V\}$
3 $\Theta \leftarrow \{0\}^{|V|\times|V|}$
4 **foreach** $v_i \in V$ **do**
5 $\quad \tau \leftarrow \min(\max(\lceil\sqrt{|\mathcal{I}_i|}\rceil, \alpha), \beta)$
6 $\quad s \leftarrow \lceil\sqrt{|\mathcal{I}_i|}\rceil$
7 $\quad$ Initialize $|\mathcal{I}_i|$ empty lists $\mathcal{L}$
8 $\quad$ **for** $t \leftarrow 1$ **to** τ **do**
9 $\quad\quad \hat{X} \leftarrow \tilde{X}$
10 $\quad\quad V^{(S)} \leftarrow s$ nodes sampled from $\mathcal{I}_i$
11 $\quad\quad V^{(R)} \leftarrow \mathcal{I}_i \setminus V^{(S)}$
12 $\quad\quad$ Set $\hat{X}^{(S)}$ to zeros
13 $\quad\quad P \leftarrow f(V, \hat{X})$
14 $\quad\quad$ Set $\hat{\boldsymbol{x}}_i$ to zeros
15 $\quad\quad P' \leftarrow f(V, \hat{X})$
16 $\quad\quad$ **foreach** $v_j \in V^{(R)}$ **do**
17 $\quad\quad\quad$ Add $\left\|\boldsymbol{p}_j - \boldsymbol{p}'_j\right\|_2$ to $\mathcal{L}_j$
18 $\quad$ **foreach** $v_j \in \mathcal{I}_i$ **do**
19 $\quad\quad \theta_{j,i} \leftarrow \text{mean}(\mathcal{L}_j)$
20 $\quad \tilde{\boldsymbol{\theta}}_{:,i} = \boldsymbol{\theta}_{:,i} / \max \boldsymbol{\theta}_{:,i}$ ▷ global attack

Algorithm 3: MAUI - Combined

Input: The inference API $f(\cdot,\cdot)$; the target nodes $V^{(T)} \subseteq V$; the threshold to run the original MAUI attack λ
Output: The influence values Θ ($\tilde{\Theta}$)

1 $\tilde{\Theta}^{(E)} \leftarrow \text{MAUI}_{(\text{E})}(f, V)$ ▷ Algorithm 2
2 $\Theta^{(C)} \leftarrow \{0\}^{|V^{(T)}|\times|V|}$
3 **foreach** $v_i \in V^{(T)}$ **do**
4 $\quad$ **foreach** $v_j \in \mathcal{I}_i$ **do**
5 $\quad\quad$ **if** $\tilde{\theta}^{(E)}_{i,j} < \lambda$ **then** ▷ for directed graph
6 $\quad\quad\quad$ **continue**
$\quad\quad$ ▷ the following is the same as Algorithm 1 ◁
7 $\quad\quad \hat{X} \leftarrow \tilde{X}$
8 $\quad\quad V^{(I)} \leftarrow \mathcal{I}_i \cap \mathcal{I}_j$
9 $\quad\quad$ Set $\hat{X}^{(I)}$ to zeros
10 $\quad\quad \boldsymbol{p}_i \leftarrow f_i(V, \hat{X})$
11 $\quad\quad$ Set $\hat{\boldsymbol{x}}_j$ to zeros
12 $\quad\quad \boldsymbol{p}'_i \leftarrow f_i(V, \hat{X})$
13 $\quad\quad \theta^{(C)}_{i,j} \leftarrow \left\|\boldsymbol{p}_i - \boldsymbol{p}'_i\right\|_2$

of GNNs and node interactions. Building upon Proposition 1, we introduce a sampling strategy to alleviate the problems of inaccurate influence estimation. To reduce the noise (i.e., the influence from other nodes) during neighborhood aggregation, we sample s nodes from $\mathcal{I}_i$ to generate the set $V^{(S)}$ and set the node features of $v \in V^{(S)}$ to zeros. Then we calculate the influence values of v_i on the remaining nodes $V^{(R)} = \mathcal{I}_i \setminus V^{(S)}$ by assigning $\hat{\boldsymbol{x}}_i$ to zeros and applying Equation (2). Since s is fixed for each node, if an unsampled node v_j is closer to v_i, their intersection of influence nodes $V^{(I)}$ will cover more sampled nodes whose features are set to zeros. This means more noise caused by other nodes in $V^{(I)}$ will be alleviated during neighborhood aggregation. Therefore, with a smaller distance, intuitively, more influence will be passed from v_i to v_j. For better accuracy, we iterate the sampling process τ times and average the results as the final influence values for every node pair.

The process of this efficient strategy is summarized in Algorithm 2. Here, we dynamically define s and τ based on the number of influence nodes to alleviate the noise fairly and effectively for each node. During implementation, we make sure the nodes are evenly sampled at line 10 and no empty lists exist in $\mathcal{L}$. Here, we adopt a sampling window to constrain the number of samples. For each target node, the amount it is sampled falls within $[\alpha, \beta]$ (line 5). The lower bound α ensures nodes with fewer influence nodes are sampled enough times; the upper bound β, the maximum sampling times for each node, ensures this strategy is efficient.

Combined strategy. To collectively enjoy the advantages of the original strategy and the efficient strategy, we combine the two strategies together. We first run the efficient strategy and obtain the normalized influence values $\Theta^{(E)}$. Later, we set a threshold in $(0, 1)$ or top-c node pair candidates, based on which, we run the original strategy. For a node pair (v_i, v_j) in an undirected graph, if $\theta^{(E)}_{i,j}$ or $\theta^{(E)}_{j,i}$ is bigger than the threshold or is the top-c candidate, its influence values $\theta^{(O)}_{i,j}$ and $\theta^{(O)}_{j,i}$ will be computed using the original strategy; otherwise, the influence values remain zeros. After completing all the valid node pairs, we obtain the normalized influence values $\Theta^{(O)}$. The ultimate influence values are computed by $\Theta^{(C)} = \Theta^{(E)} + \Theta^{(O)}$. Note that we do not apply another normalization to $\Theta^{(C)}$. The algorithm is presented in Algorithm 3.

Complexity analysis. The forward operation of GNNs is the most time-consuming process in our attacks, but it is constant. Thereby, for time complexity analysis, we consider the complexity of GNN operation as a unit, i.e., each API query takes $O(1)$. According to Algorithms 1 to 3, each attack iteration takes exactly two API queries.

MAUI is an exhaustive attack strategy where the complete set of influence nodes must be traversed for each target node. Suppose the target graph contains n nodes and m undirected edges, and each node is connected to $2m/n$ edges on average. The average number of influence nodes is estimated to be:

$$|\mathcal{I}_i| \leq \sum_{l=1}^{L} 2m/n \cdot (2m/n - 1)^{l-1} \leq n - 1. \tag{4}$$

We assume the target node has $2m/n$ 1-hop neighbors and each neighbor connects to $2m/n - 1$ new neighbors. So there are $2m/n \cdot (2m/n-1)^{l-1}$ nodes that can be reached by an l-hop path from the target node. After l layers of GNNs, considering the interconnections, the number of influence nodes is no bigger than $\sum_{l=1}^{L} 2m/n \cdot (2m/n-1)^{l-1}$. The worst case is when $|\mathcal{I}_i| = n - 1$. According to Algorithm 1, the time complexity of MAUI is $O(n \cdot (2m/n)^L)$, with the worst case being $O(n^2)$. In this work, we only consider sparse

Table 3: Statistics of datasets.

Dataset	# nodes	# edges	# features	# classes
Cora	2,708	5,278	1,433	7
Citeseer	3,327	4,552	3,703	6
Coauthor-CS	18,333	81,894	6,805	15
Facebook	22,470	171,002	128	4
Github	37,300	289,003	128	2
LastFM-Asia	7,624	27,806	128	18

graph structures ($|\mathcal{I}_i| \ll n-1$) as they are the common scenario in the real world and more meaningful to explore [42].

Similarly, the time complexity of the efficient strategy is $O(n \cdot \tau)$, where τ should be significantly smaller than the average number of influence nodes. In Algorithm 2, the average of τ by definition is $\sqrt{|\mathcal{I}_i|}$. Therefore, the time complexity is $O(n \cdot (2m/n)^{L/2})$, which does not exceed $O(n \cdot \beta)$. Here, the costs of lines 16–19 can be safely disregarded.

The time complexity of the combined strategy is not easily determined, as how many node pairs are further estimated by the original strategy depends on the threshold and the previous results. Nevertheless, a feasible threshold can be determined based on the previous results to ensure the time complexity is substantially reduced compared to that of the original strategy.

5 EXPERIMENT

5.1 Experimental Setup

We conduct our experiment on a server with two Intel Xeon Gold 6126 CPUs @ 2.60GHz, each of which has 12 cores and 24 threads. The implementation in this work is available for reproduction[3].

Datasets. In this work, we use six public datasets of node classification tasks.

- *Citation network:* Cora and Citeseer are similar citation networks [70], where the nodes are documents with bag-of-words represented by node features, and edges represent citation links. The task is to predict the documents' research fields.
- *Coauthor network:* Nodes in Coauther-CS [50] represent authors with node features as keyword representations from their papers. The nodes are connected if the corresponding authors have a coauthor relationship. The classes are the fields of study.
- *Web network:* Facebook Page-page network [47] is derived from verified Facebook sites. Each node is an official Facebook page and an edge is mutual likes between pages. Node features are encoded from site descriptions. Each site is categorized based on its content topic.
- *Social network:* Github [47] and LastFM-Asia [48] are similar social networks collected from the corresponding services. They contain nodes as users and edges as mutual follower relationships. The node features are extracted from user information. The task is to classify users' occupational fields for Github and users' nationalities for LastFM-Asia.

Models. We use three GNNs with different structures as the target models: a 2-layer Graph Convolutional Network (GCN) [29], a 3-layer GCN (in Appendix B), a 2-layer Graph Attention Network (GAT) [54], and a 2-layer GraphSAGE [20]. For model training, all the datasets are randomly split into the train, validation, and test sets with ratios of {0.6, 0.2, 0.2}. We choose 2-layer GCNs to be our major target model, as they are the most classic and widely used GNNs.

All GNN models are implemented with Pytorch-Geometric [15]. Other than the GNN structures, the hyperparameters of GNNs are uniformly set. We adopt the commonly used settings to the hyperparameters and GNN structures. We assign the number of units in a hidden layer to 64. We adopt ReLU as the activation function for the hidden layers and Softmax for the output layer. To reduce overfitting, we use dropout in each hidden layer with a rate of 0.5. The learning rate is set to 0.01. The number of heads is 8 for GATs.

Baselines. We compare Maui with two closely related attacks: Link Stealing Attack-2 (LSA2) [21] and LinkTeller [62]. LinkTeller, which also attacks vertically partitioned GNNs, assumes that the attacker has knowledge of node features. It then makes minor perturbations to one node's features and measures the effect on all other nodes — they use this as that node's influence value. It differs from Maui in its assumption that node features are known, and in its assumption that one node having a strong influence on another directly correlates to the likelihood of an edge between them. Whereas, we compare influence values per target node, as described in Section 4.3. We consider two strategies for LSA2: LSA2-feat, and LSA2-post. LSA2-feat directly compares node features, assuming that nodes with similar features will have edges in the graph. LSA2-post compares the posteriors of features, using the distance between GNN outputs for given nodes as the likelihood of an edge existing between them. Both LSA2 and LinkTeller require knowledge of the real node features and predictions. Therefore, the baseline attacks utilize more information than Maui. The baseline attacks are implemented from the source code shared by Wu et al. [62]. Additionally, we employ a decoder as a white-box graph reconstruction attack [12]. Intuitively, nodes with similar embeddings from GNNs tend to be connected. This approach takes the node embeddings and decodes them into an adjacency matrix with continuous values representing the influence values.

Metrics. We evaluate the attack performance with average precision (AP) as the major metric. The edges in a real-world graph are usually sparse, i.e., the positive labels of ground truth are significantly imbalanced. The target of the attackers is to infer the connected node pairs, the number of which is considerably smaller than the quantity of all the node pairs. Furthermore, the outputs of the attackers are continuous values representing the distance/influence of node pairs, so the attackers have to pick the node pairs with the largest influence values as the positive predictions. In summary, we consider AP the best option to represent the correctness of edge inference for the following reasons: 1) the labels of ground truth are not balanced; 2) we favor the positive predictions and the positive ground truths; 3) the threshold for classification is not deterministic. Specifically, AP is calculated as: $\mathrm{AP} = \sum_t (\mathrm{Re}_t - \mathrm{Re}_{t-1})\mathrm{Pre}_t$, where Re_t represents the recall at the t-th threshold. For deeper analysis, we also consider precision at k (p@k) and recall at k (r@k), where we choose the top-k results and label them as positive, then calculate the corresponding precision and recall.

[3] https://github.com/iHeartGraph/Maui

Table 4: Performance of local attack with 2-layer GCNs. We show the average precision (AP). Note the results on Cora and Citeseer are deterministic, so the SD is 0.

Attack	Cora	Citeseer	Coauthor-CS	Facebook	Github	LastFM-Asia
Decoder	51.5	68.8	75.0±1.6	63.6±1.0	58.0±1.6	49.3±0.8
LSA2-post	45.2	64.2	70.6±1.0	59.1±0.6	31.2±7.2	48.1±1.3
LSA2-feat	45.6	64.3	88.6±0.8	49.6±1.0	31.2±6.5	47.0±0.6
LinkTeller	77.7	86.8	88.2±0.6	83.0±0.6	85.4±3.2	81.1±0.6
MAUI	**93.3**	**95.0**	**99.4±0.1**	**95.5±0.2**	**95.1±1.3**	**96.0±0.2**

Attack Types. We evaluate the attacks on both local and global attacks. For local attacks, the influence values are the original attack outputs, specifically Algorithm 1 for MAUI. We only calculate the influence values between the target nodes and their influence nodes rather than for all the node pairs. As a result, the evaluation of LSA2 and Decoder is better than what was originally reported. For global attacks, we transfer the local influence values into global influence values by normalization (Equation (3)). As this leads to better performance, unless otherwise specified, evaluations of the global attacks are results with normalization. Hence, the results we report for baseline attacks are better than the original, non-normalized methods.

Specifications. In our experiments, we use entire graphs as the target graphs for citation networks (Cora and Citeseer). For the remaining larger datasets, for the ease of evaluation, we evaluate the attacks on a randomly sampled subgraph containing 3,000 nodes and repeat five times. We use deterministic random sampling to ensure different attacks perform on the same target subgraphs. We report the means and the standard deviations (SDs) for these datasets. We assume the inference API does not include feature processing stage (e.g., feature normalization), i.e., the node features we provide will be directly fed into the GNN model. However, only LinkTeller is negatively affected by feature normalization, which we analyze in Section 5.6.

All the attacks are performed with their corresponding capabilities. For example, if node features are necessary for an attack (such as LinkTeller), then the attack has access to node features. In Appendix D.2, we also evaluate the performance of MAUI using the original node features.

5.2 Local Attack Evaluation

We conduct the local attack with 2-layer GCNs, obtain the AP for each node, and report the average of each dataset in Table 4. We observe that MAUI significantly outperforms the other baseline approaches.

Our experimental results demonstrate that node similarity does not always guarantee a connection, as evidenced by the performance of Decoder and LSA2. LSA2, which does not consider GNN interactions or graph properties, achieved the lowest APs overall. This attack will fail on specific graphs where the connected nodes belong to different types, such as molecule graphs and bioinformatics graphs [43], and synthetic graphs [75] where all the node features are the same. In contrast, LinkTeller, which leverages node interactions through GNN computation and estimates influence using gradients, achieves the best performance among all prior works tested. However, LinkTeller does not thoroughly consider interconnections between nodes and relies on extra knowledge of node features, which explains why it is not the best-performing method.

Table 5: Performance of global attack with 2-layer GCNs.

Attack		Cora	Citeseer	Coauthor-CS	Facebook	Github	LastFM-Asia
w/o norm	Decoder	15.1	28.0	43.1±2.4	28.1±1.1	20.1±3.2	23.9±0.6
	LSA2-post	15.7	34.4	49.5±2.0	27.2±1.3	5.4±2.5	22.6±1.3
	LSA2-feat	22.0	36.7	**74.8±3.1**	20.3±1.9	6.3±2.9	22.3±1.4
	LinkTeller	35.4	61.2	52.6±1.0	44.4±0.5	43.0±4.6	45.8±1.0
	MAUI	**78.9**	**73.5**	73.2±1.1	**63.3±2.0**	**67.2±5.2**	**73.8±1.2**
w/ norm	Decoder	31.3	42.3	53.0±2.7	37.2±1.7	29.4±4.1	27.8±0.7
	LSA2-post	19.7	35.1	49.1±1.2	30.2±0.9	6.7±3.1	24.2±1.1
	LSA2-feat	23.2	39.1	76.2±1.9	25.8±1.0	9.1±3.8	24.5±0.9
	LinkTeller	62.3	73.5	75.4±1.4	67.0±0.8	62.7±9.8	65.3±0.9
	MAUI	**87.3**	**84.2**	**96.2±0.7**	**85.5±0.5**	**83.4±3.2**	**87.4±0.6**

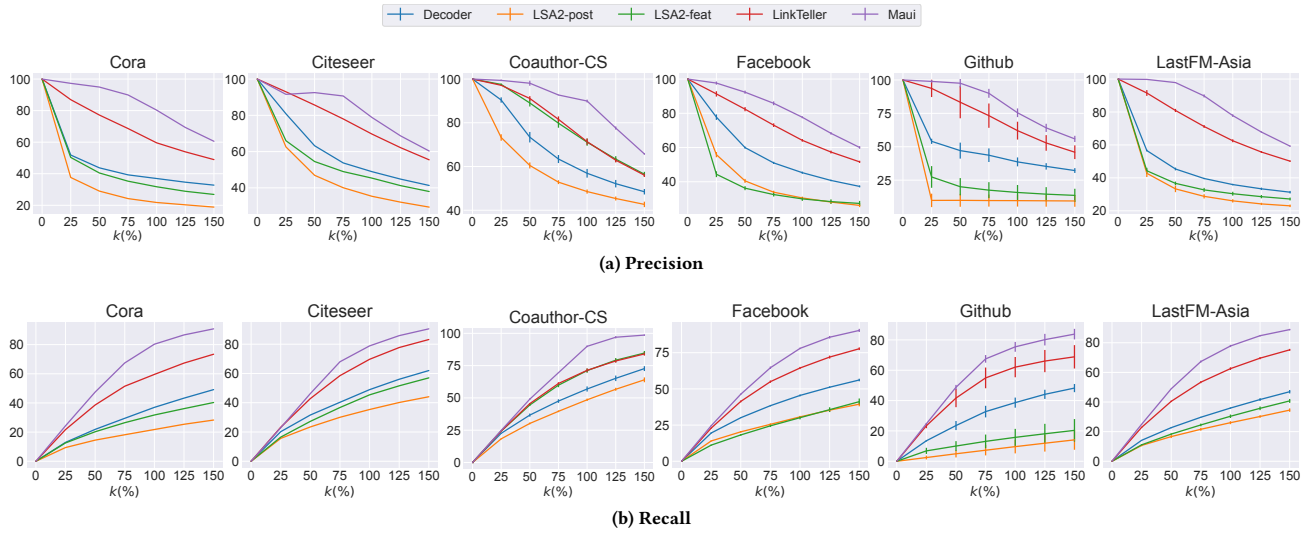
Maui is the most effective method; in particular, MAUI outperforms LinkTeller by 12.0% on average. Considering we do not require extra knowledge other than the inference API, MAUI greatly improves the link inference attacks. We collectively consider the computation process of GNNs and the graph structure for influence value estimation. Moreover, we set all the node features uniformly to prevent inaccurate inference by feature interactions. The result demonstrates the lack of node feature knowledge does not degrade the attack performance.

We observe that the attacks tend to perform better when the dimension of node features and predictions is relatively large. Predictions with larger dimensions present stronger expressive power and are more sensitive to changes in node features. Nonetheless, while prior works fail on datasets with fewer classes, e.g., Github, MAUI remains effective. Thus, we conclude that GNN predictions alone contain rich information about the graph structure, which presents serious concerns about privacy leakage.

5.3 Global Attack Evaluation

Attack without normalization. First, we calculate the AP for the whole graph using the same influence values from the local attack. The results are collected in the upper section of Table 5. They suggest that MAUI overall still outperforms the other attacks, but due to the lack of extra information, especially the degree of each node, performance declines sharply compared to the local attacks. Without enough knowledge, we cannot directly use the local influence values on global attacks. Even with extra knowledge, since the precision of local attack is not 100%, the performance of global attack is not well guaranteed. The huge decline firmly indicates the incompatibility of the influence values from a global perspective. Particularly, a larger influence value does not guarantee a better chance of connection.

Attack with normalization. To address the above challenge, we apply min-max normalization to the influence values of each node and obtain the AP for the whole graph. To effectively demonstrate the advancement of normalization, we adopt it to the baseline methods as well. Note the original baseline methods do not contain the normalization process. The results show that normalization offers great improvements in the global attack. In Cora dataset,



(a) Precision

(b) Recall

Figure 3: Precision and recall at different top-k. We vary the top-k ratio of the predicted edges to the real edges, ranging from 25% to 150%, then calculate the corresponding precision and recall.

normalization has improved the performance of MAUI by 8.4% and LinkTeller by 26.9%. These improvements are from the global incompatibility of unnormalized influence values, rather than the attack performance or the lack of further knowledge.

It is expected that the performance of the global attack is lower than that of the local attack, given the greater challenge of inferring edges across the entire graph. Under the most difficult setting, we can only normalize the influence values to ensure global compatibility, and we assume that node pairs with the top influence values are connected. We observe that this approach is less effective on graphs with irregular edge density. In particular, graphs with high density tend to have more interactions between nodes, which leads to more globally incompatible influence values even after normalization, and results in a greater decline in performance for the global attack. With extra knowledge such as node degrees, MAUI is able to achieve better results, which we show in Appendix D.1.

Precision and recall. We vary the top-k ratio of the predicted edges to the real edges, ranging from 25% to 150%, then calculate the corresponding precision and recall. The ideal precision is $\min(1, 1/k)$ and the recall is expected to be $\min(k, 1)$. The precision and recall at top-100% are considered as the attack accuracy. We report the results in Figure 3.

In general, MAUI achieves the best results and largely outperforms others. The results of MAUI are relatively stable, especially in the Github dataset, because MAUI obtains the influence values for every node pair individually. The graph of Github is uneven, i.e., the subgraphs are either dense or sparse, which leads to the unstable attack performance of other attacks. It is also evident that methods actively using the feedback of GNNs achieve better results. Overall, there exists a large gap between LinkTeller and LSA2-feat.

Finally, we observe that MAUI is more sensitive to the correct node pairs at the lower top-k. From 50% to 75%, the precision result in Figure 3a is flatter than prior works, while the recall curve is steeper in Figure 3b. According to our assumption, 1-hop neighbors with more connections tend to have larger influence values. As more node pairs are discovered, MAUI will primarily compare nodes between 1-hop neighbors with fewer connections and 2-hop neighbors that are largely connected, both of which tend to obtain higher influence values than the rest.

Table 6: Performance with 2-layer GATs.

Type	Attack	Cora	Citeseer	Coauthor-CS	Facebook	Github	LastFM-Asia
Local	Decoder	37.8	58.7	65.0±1.2	53.1±0.8	37.4±7.8	38.2±1.2
	LSA2-post	49.9	72.1	76.8±1.4	63.2±0.4	32.1±7.0	55.4±1.1
	LSA2-feat	45.6	64.3	88.6±0.8	49.6±1.0	31.2±6.5	47.0±0.6
	LinkTeller	81.7	88.4	91.8±0.4	82.6±1.1	92.1±0.3	83.1±0.7
	MAUI	**97.0**	**95.0**	**99.1±0.2**	**94.9±0.2**	**98.3±0.2**	**97.4±0.2**
Global	Decoder	18.7	28.2	36.8±1.5	23.2±1.5	11.0±4.2	15.7±0.7
	LSA2-post	21.4	43.8	55.7±2.3	33.3±1.4	7.1±3.4	27.4±1.4
	LSA2-feat	23.2	39.1	76.2±1.9	25.8±1.0	9.1±3.8	24.5±0.9
	LinkTeller	70.8	78.0	83.4±1.0	68.3±1.6	82.6±1.1	72.7±1.0
	MAUI	**92.6**	**86.7**	**97.3±0.6**	**88.2±0.4**	**91.9±1.1**	**91.9±0.7**

5.4 Compatibility Evaluation

We evaluate the attacks with 2-layer GATs and 2-layer GraphSAGE to demonstrate the compatibility of MAUI with other GNNs. We report the AP of both local attack and global attack in Tables 6 and 7.

2-layer GATs. All the attacks have maintained their performance and MAUI continues to outperform the other attacks. Compared with the results of the 2-layer GCNs in Table 4, the attacks overall achieve better results with GATs except for the Decoder, due to the different model architecture. The results of LSA2-feat are unchanged since it only measures the similarity of node features, which is irrelevant to GNN models. We compare the attack performance of different models mainly from the local experiments

Table 7: Performance with 2-layer GraphSAGE.

Type	Attack	Cora	Citeseer	Coauthor-CS	Facebook	Github	LastFM-Asia
Local	Decoder	39.7	61.9	63.0±1.2	55.0±1.5	38.8±8.0	42.1±1.1
	LSA2-post	45.8	66.2	**63.4±1.4**	55.3±1.2	27.4±6.0	47.5±1.4
	LinkTeller	79.4	84.9	44.4±2.6	67.6±1.0	84.4±0.7	60.5±1.3
	Maui	**100.0**	**100.0**	41.2±1.0	**98.9±0.2**	**100.0±0.0**	**100.0±0.0**
Global	Decoder	18.1	32.3	38.5±2.0	28.5±1.0	12.7±5.3	20.0±0.8
	LSA2-post	18.1	36.6	40.9±1.6	27.1±0.7	6.5±3.1	22.2±1.1
	LinkTeller	70.8	76.8	27.1±2.8	49.4±1.3	80.8±3.7	45.2±2.1
	Maui	**100.0**	**100.0**	**55.1±2.1**	**99.6±0.1**	**100.0±0.0**	**100.0±0.0**
Model Accuracy		*85.3*	*73.1*	*92.7*	*94.1*	*85.9*	*84.6*

because the results from the local attack are calculated using the original algorithm's output (Algorithm 1 without global normalization) and they reflect the overall attack performance.

2-layer GraphSAGE. Additionally, we evaluate different attacks with 2-layer GraphSAGE (max-pooling as the aggregation method) and report the results in Table 7. Overall, the performance drops from attacks on other GNN architectures, while Maui still stays on top. Nodes with smaller distances tend to receive more messages from each other, which leads to a higher probability of retaining the messages by max-pooling. However, this action is non-differentiable and a slight change in node features can lead to a different message sampling in each layer. This can largely damage the effectiveness of LinkTeller. Max-pooling is not commonly used in GNN layers due to information loss, but it can be effective as a defense. Due to the max-pooling aggregation, information from one node may be passed through multiple edges without critical loss, so the influence of a node pair is retained. The attack performance of Maui highly depends on node feature initialization, which may require a lot of testing to attack GraphSAGE.

Besides GNNs with different aggregation methods, we conduct further experiments with 3-layer GCNs in Appendix B. These experiments have similar results to those shown here, with Maui remaining the most effective attack.

5.5 Running Time Evaluation

All experiments are executed on CPUs, which process sequentially without additional memory overhead, to ensure a fair comparison. Our implementation strictly follows Algorithm 1, which operates sequentially. However, a GPU implementation could concurrently execute the for-loop in lines 5-12 of Algorithm 1 by duplicating the graphs and running the model with all the duplications as input. This approach might mirror simultaneous API calls in practical applications. Nonetheless, the baseline methods we evaluate cannot trivially be distributed. Consequently, it would be unfair to compare the running time of our approach (which executes in parallel with additional resources) to the running time of the baseline methods (which execute sequentially). We report the running time of attacks with 2-layer GCNs in Table 8. Additionally, we report the running time to identify the influence nodes, which is necessary for all the attacks. This process only requires a single execution, after which the results can be stored for subsequent utilization. The running time with 3-layer GCNs can be found in Appendix B. The decoder, which is only used as a baseline for attack performance, is omitted in the run-time evaluation. The influence values for all the node pairs are generated in $O(1)$, with only constant-time call to the GNNs.

Table 8: Running time (in seconds) with 2-layer GCNs.

Attack	Cora	Citeseer	Coauthor-CS	Facebook	Github	LastFM-Asia
LSA2-post	13.0	14.7	11.3±0.1	11.6±0.1	22.3±14.8	18.6±2.2
LSA2-feat	12.2	14.8	12.5±0.1	11.6±0.1	19.5±10.1	15.5±0.3
LinkTeller	**11.85**	**9.15**	**4.4±0.3**	**4.9±0.2**	**6.7±2.2**	**9.0±0.6**
Maui	348.6	229.91	66.7±7.9	109.8±11.0	828.2±1280.4	212.2±21.5
$\mathcal{I}$	*20.7*	*40.4*	*30.5±2.2*	*15.8±0.1*	*11.4±1.0*	*18.8±0.9*

We observe that Maui is the most time-consuming of all the attacks, while LinkTeller achieves the smallest running time. For the Github dataset, there is a densely connected subgraph out of the five samples in our experiment, which leads to a huge number of node pairs for Maui to estimate. However, the huge standard deviation (even larger than the mean value) indicates that Maui still performs in reasonable running time on other subgraphs, considering Maui estimates each node pair individually for better inference precision.

The time complexity of LSA2 is the same as Maui since we only consider influence nodes rather than all the node pairs. However, LSA2 only calculates the node similarity without running the GNN model. In this experiment, the time complexity of LinkTeller is $O(n)$. In each round, the attack calls the inference API once and collects all the influence values of one node. The small running time is obtained because 1) the graph is relatively small, and 2) the influence nodes of target nodes are overlapped. Due to the first reason, we can feed the whole graph to a GNN model and obtain the influence values from one node to all. For the second reason, the target nodes are each others' influence nodes, so running LinkTeller n times is enough to infer the edges for the whole graph. Consider a large graph where the influence nodes of the target nodes are not overlapped, the time complexity of LinkTeller will be the same as Maui. Our experiment can be considered as attack on a cluster of connected nodes.

Considering the significant advancement we have achieved without the knowledge of node features, it is reasonable that we demand more running time to estimate the influence values. Especially considering that in a real attack, it may not be necessary to infer the edges for the whole graph. Maui is conducted on all the candidate node pairs, while in practice, we can filter out a big portion of redundant node pairs with various strategies such as our efficient strategy. Additionally, the running time can be largely reduced by using GPUs with simultaneous execution.

5.6 Robustness Evaluation

We evaluate the robustness of attacks under two defense mechanisms, LapGraph [62] (proposed by LinkTeller) and our proposed mechanism, EdgeAttn. Following the defense mechanism, we employ 2-layer GCNs as the target models in order to compare with the previous results and obtain the results of the local attack and the global attack. Additional details of both mechanisms can be found in Appendix C. It is noteworthy that, although research has been conducted to safeguard GNN privacy [78], few studies have

Table 9: Performance with 2-layer GCNs under LapGraph. According to the analysis from Wu et al. [62], we set ϵ to 10.

Type	Attack	Cora	Citeseer	Coauthor-CS	Facebook	Github	LastFM-Asia
Local	Decoder	45.9	57.1	63.3±1.4	55.3±0.9	49.2±1.4	44.2±1.0
	LSA2-post	41.4	57.2	62.9±1.2	52.0±1.0	25.5±5.4	44.0±1.0
	LinkTeller	66.8	72.9	71.7±0.7	71.5±1.0	66.7±2.2	68.9±0.8
	Maui	**84.5**	**81.5**	**81.7±0.9**	**83.0±0.6**	**72.5±1.3**	**80.6±0.8**
Global	Decoder	27.6	34.5	52.9±1.7	28.6±1.4	22.1±3.1	23.9±0.7
	LSA2-post	19.1	31.6	41.4±1.2	25.5±1.0	6.4±2.9	22.6±1.0
	LinkTeller	51.3	58.3	53.4±1.1	51.7±1.3	39.4±4.0	50.7±0.9
	Maui	**77.7**	**71.1**	**69.5±1.2**	**66.0±0.3**	**50.8±0.7**	**67.9±0.9**

Table 10: Performance with 2-layer GCNs under EdgeAttn.

Type	Attack	Cora	Citeseer	Coauthor-CS	Facebook	Github	LastFM-Asia
Local	Decoder	39.2	57.9	65.2±1.3	53.0±0.9	35.5±7.7	38.5±1.2
	LSA2-post	51.5	71.6	76.6±1.3	**57.9±0.6**	31.5±7.4	53.9±1.0
	LinkTeller	36.9	57.3	62.2±1.3	51.8±0.9	37.2±8.3	40.3±0.8
	Maui	**78.7**	**86.8**	**93.4±0.3**	56.3±0.6	**86.2±3.1**	**88.2±0.5**
Global	Decoder	18.0	28.7	37.9±1.5	24.2±1.8	10.9±3.8	16.2±0.8
	LSA2-post	20.2	41.7	55.2±2.1	28.6±2.0	6.8±3.2	25.7±1.1
	LinkTeller	15.1	28.2	36.9±1.7	23.9±0.8	9.7±4.5	17.4±0.8
	Maui	**60.8**	**73.8**	**78.2±0.8**	**29.0±0.7**	**66.3±8.1**	**75.2±1.4**

delved into the preservation specifically against link inference attacks. Given that privacy-preserving techniques can shield GNNs from various types of attacks, we have opted to assess the effectiveness of two commonly used techniques—differential privacy (LapGraph) and adversarial training (EdgeAttn). Additionally, we find LinkTeller can be vulnerable to model manipulations (e.g., adding feature normalization). The attack can be invalidated by adding a simple layer.

LapGraph. LapGraph is an edge perturbation mechanism that guarantees ϵ-edge differential privacy (DP) [13] while preserving the density of the target graph. LapGraph randomly perturbs the edges of the entire dataset with a privacy budget ϵ, before training and inference. To extend the metaphor from Section 1, Alice first employs LapGraph on the original graph, then releases the perturbed graph on the ML platform. Next, a GNN model will be trained on the perturbed graph. At the inference stage, the same perturbed graph is used for prediction.

In comparison with Tables 4 and 5, the results in Table 9 show that LapGraph generally alleviates the privacy leakage against the link inference attacks, while Maui still outperforms other attacks. We do not show the results of LAS2-feat because it only measures the similarity of node features, so it is unaffected. LapGraph performs the best on Github dataset because it perturbs more edges in the dense subgraph.

These results suggest that the privacy concern is not well alleviated under this defense mechanism, although the attacker cannot be confident in their attack performance, as it is unclear which edges overlap with the true data. This defense mechanism protects graph information through input perturbation on graph structure. The perturbation guidance is only within the input space without GNN feedback. To maintain GNN performance, only slight perturbation of the graph structure is acceptable. We believe that with additional API calls, the majority of edges can be successfully identified through overlapping results. Furthermore, the graph structure from certain datasets cannot be randomly perturbed because it contains important information, such as chemical graphs [2], transaction graphs [57] and provenance graphs [28] with strict rules about edges.

Table 11: Performance of LinkTeller when applying feature normalization. We show [the raw AP | Δ AP] in each cell.

Dataset	GCNs		GATs	
	local	global	local	global
Cora	23.4 | -54.3	9.1 | -53.2	25.8 | -55.9	10.2 | -60.6
Citeseer	42.6 | -44.2	17.1 | -56.4	43.7 | -44.7	19.0 | -59.0
Coauthor-CS	56.3±1.0 | -31.9	34.2±1.9 | -41.2	56.9±1.2 | -34.9	35.1±1.7 | -48.3
Facebook	39.1±1.0 | -43.9	15.6±0.8 | -51.4	39.3±1.6 | -43.3	15.3±0.9 | -53.0
Github	17.7±3.6 | -67.7	3.9±1.8 | -58.8	21.1±2.2 | -71.0	3.5±1.9 | -79.1
LastFM-Asia	43.1±0.9 | -38.0	16.3±0.8 | -49.0	44.0±0.7 | -39.1	16.1±0.7 | -56.6

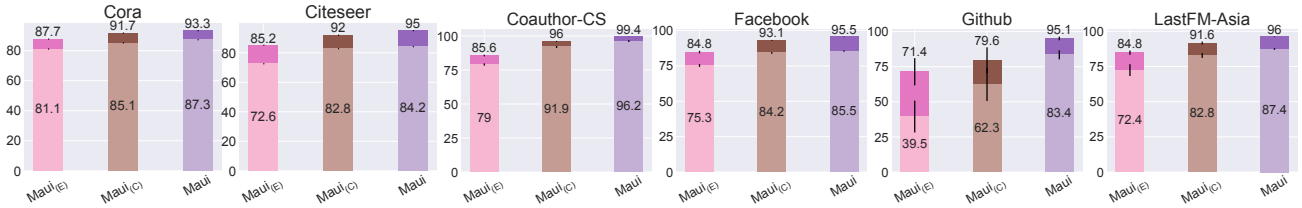
If we take the perturbed graphs as ground truth, the evaluation is considered the same as the evaluation with 2-layer GCNs, which we show in Appendix C.

EdgeAttn. Different from LapGraph, EdgeAttn is trained on the original graph. It manipulates edge attention to prevent the information from passing evenly to neighbor nodes. Similar to GATs, the attention for each edge is learned through the training process, where the edges contributing to the prediction tend to gain more attention. We add noise to the attention values in each GNN layer to prevent overfitting and improve the robustness.

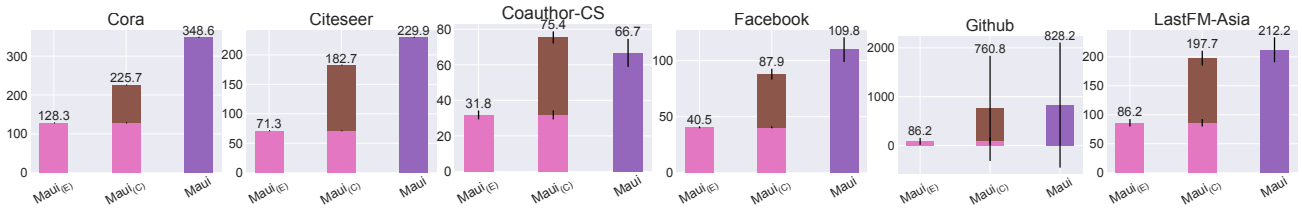
Table 10 shows the performance of LinkTeller drops significantly under EdgeAttn. Although EdgeAttn increases the prediction similarity between neighbor nodes, it preserves edge privacy from attacks that require the feedback of GNNs. The performance of Maui does not greatly decline due to the full utilization of graph structure. It can be observed that EdgeAttn works better in relatively dense graphs where edge attention plays an important role in the predictions. For this reason, EdgeAttn successfully protects edge privacy in Facebook dataset, but the attacks still maintain their performance in Coauthor-CS dataset. This dataset has lower node degrees on average, and EdgeAttn is trained on the original graph, so it is more difficult to hide the graph structure from Maui. However, a slight manipulation of edge attention can significantly decrease the performance of LinkTeller.

While EdgeAttn alleviates the limitations of LapGraph, privacy is still not well protected. The condition for privacy protection is rigid. The graphs must be dense enough to obfuscate individual edges, and the predictions should largely rely on edge attention. In this case, the learned edge attention can be highly imbalanced so certain edge attention values can be too small for information to pass through the corresponding edges.

Model Manipulation against LinkTeller. We obtain the results of LinkTeller when feature normalization is applied to the 2-layer GNNs in Table 11. The results have hugely declined from the original. Due to LinkTeller's gradient estimation strategy, it is easy to be manipulated, e.g., by feature normalization. This is not a defense mechanism but is commonly used in real life to make sure the model parameters are well learned during training. Feature



(a) AP. The light and the dark bars represent the global and local attack performance, respectively.



(b) Running time (in seconds). For MAUI$_{(C)}$, we present the time for MAUI$_{(E)}$ and MAUI separately, and label the total running time.

Figure 4: Performance of MAUI with different estimation strategies.

normalization prevents LinkTeller from accurately estimating the gradients, thus destroying its effectiveness. Other operations, such as layer/batch normalization can be considered not only to improve the performance but prevent privacy attacks like LinkTeller. However, they do not affect the other attacks, which indicates the urgent need of countermeasures.

5.7 Estimation Strategy Evaluation

We follow Algorithm 2 and use dynamic sampling for the efficient strategy, denoted as MAUI$_{(E)}$. We set the threshold to 0.2 for the combined strategy, MAUI$_{(C)}$, to balance the running time and attack performance. The attacks are conducted with 2-layer GCNs. We evaluate the estimation strategies by the attack performance and running time. The result is reported in Figure 4.

Attack performance. We present the attack performance in Figure 4a. In general, the results become better as the original estimation strategy is more involved in the computation. MAUI$_{(C)}$ achieves better results than MAUI$_{(E)}$, while underperforming the original MAUI. The use of MAUI greatly improves the performance of MAUI$_{(E)}$. Specifically, MAUI$_{(C)}$ improves the performance of MAUI$_{(E)}$ by 7.4% in local attack. Other than Github dataset, the APs MAUI$_{(C)}$ obtains are over 90%, which are more approximate to MAUI than MAUI$_{(E)}$. With relatively dense subgraphs, MAUI$_{(E)}$ shows lower AP and high standard deviation due to the limited sampling times τ and the infeasible quantities of sampled nodes s. Compared with Table 4, the estimation strategies overall outperform the baseline methods.

Running time. The running times of different strategies are shown in Figure 4b. MAUI$_{(E)}$ greatly improves the running time in contrast to MAUI$_{(C)}$ and MAUI. MAUI$_{(C)}$, as a combined strategy, outperforms MAUI$_{(E)}$ in terms of precision, and alleviates the time cost of MAUI, especially with a dense graph or a deep GNN model. The reason why the running times of MAUI$_{(C)}$ and MAUI do not differ greatly in several datasets is that the experiment is conducted with 2-layer GCNs and the graphs are sparse, e.g., Coathor-CS dataset. With 3-layer GCNs, MAUI$_{(C)}$ will significantly reduce the running time for MAUI, and the result can be found in Appendix B. Since this strategy is dependent on MAUI$_{(E)}$, which does not require a long running time, we can allocate more computation to MAUI$_{(E)}$ for better performance. That is, we can increase the sampling times for each target node. As the performance of MAUI$_{(E)}$ is refined, we can choose a higher threshold to conduct the original estimation strategy. The other way to improve MAUI$_{(C)}$ is to use a different base method, e.g., LinkTeller. Overall, MAUI$_{(C)}$ and MAUI are the optimal choices when only a limited number of target nodes are considered.

6 RELATED WORK

Graph Neural Networks. Graph Neural Networks (GNNs), as deep learning models for graph data, were first motivated from recurrent neural networks and introduced by Gori et al. [19]. Inspired by the concept, Bruna et al. [3] took a further step to adapt the convolutional operation from grid data to graph data and developed Graph Convolutional Networks (GCNs) based on spectral graph theory. The spectral-based GNNs were then improved by various works [9, 29, 31]. To overcome the expensive consumption of using the full adjacency matrix for spectral-based GNNs, new, spatial-based GNNs such as GraphSAGE [20], GATs [54] and GINs [67], which adopt the neighborhood aggregation scheme, were developed. Building upon these models, countless works explored GNNs' utility for in a variety of problem domains, including node-level [5, 41], edge-level [27, 77], and graph-level tasks [14, 74]. Other than general node classification, link prediction, and graph classification tasks, GNNs have proven the ability to handle tasks like graph generation [52] and graph clustering [53]. Because of their remarkable advancement, GNNs have been widely deployed across different domains to serve our daily life: in fields such as computer vision [11, 34], natural language processing [71], recommendation systems [73], and many others [24, 58].

Privacy of GNNs. Attacks on GNNs target two major categories: GNN models [51, 60] and graph data [79]. The model-targeted attacks aim to infer the model architectures or parameters. The private information of graph data contains not only the graph components [80] but also the graph properties [79] (e.g., the graph size) and training membership [12] (e.g., the nodes used for training).

Zhang et al. [76] generalize the existing privacy attacks into three major categories: *model extraction attacks*, *membership inference attacks*, and *model inversion attacks*. Model extraction attacks aim to construct a surrogate model by stealing the information of a GNN model. Recent works [51, 60] have investigated model extraction attacks with different levels of capacities, e.g., whether the graph structure is missing. Membership inference attacks aim to infer whether a target component is used to train the target model, which has been studied for node classification [22, 44] and graph classification tasks [56, 59]. Model inversion attacks aim to infer the target attributes from the corresponding outputs of GNN models, which can be further divided into reconstruction attacks and property inference attacks. Reconstruction attacks, a.k.a. attribute inference attacks, are mostly studied on the graph structure under both white-box [12, 80] and black-box settings [21, 33, 62]. Including LSA2, He et al. [21] has proposed eight Link Stealing Attacks in total with different capabilities, which include whether an attack has access to node features, partial graph, and a shadow dataset. Zhang et al. [79] take the initial effort to infer the properties of a graph (e.g., the graph density) by its embeddings.

Although the defense against privacy attacks is not extensively studied, there are four major techniques for privacy-preserving GNNs [8, 78]: *differential privacy, federated learning*, and *adversarial privacy-preserving* and *latent factor disentangling*. Differential privacy [13] guarantees an attack cannot infer the information of a single sample. Federated learning [4, 61] allows individuals to train a GNN model collectively without sharing the original data. Adversarial privacy-preserving [23, 55] prevents sensitive information from leakage by adopting an adversarial function. Latent factor disentangling [32] disentangles sensitive information from the embedding space.

7 DISCUSSION

Limitations of Maui. Our work primarily focuses on node classification tasks, but with additional knowledge, such as node embeddings, the attack can be adapted to a broader range of tasks, including graph classification. This adaptation involves replacing node predictions with node embeddings, which are more expressive and could potentially enhance performance in node classification tasks.

Compared to other attacks, there is a tradeoff between computation time and attack performance in Maui. To achieve better performance without additional knowledge, Maui estimates influence values for each individual node pair. This approach can be computationally expensive. To address this limitation, we have developed different estimation strategies to strike a balance between time consumption and attack performance.

Another limitation of Maui is that it needs to conduct global attacks on a subgraph to determine the existence of edges between node pairs. The threshold for distinguishing edge existence is not deterministic, as the influence values indicate the influence of one node on another, rather than the probabilities of connections. This requires attackers to collect sufficient outputs and identify node pairs with larger influence values. Additionally, the knowledge of node degrees is not available, which would eliminate the need for a global attack.

Despite these limitations, Maui significantly outperforms state-of-the-art approaches, particularly in challenging settings. Our experiments demonstrate that other attacks fail to achieve better performance even with extra knowledge. To the best of our knowledge, there is no task where other attacks succeed while Maui fails under these challenging conditions.

Possible defense strategies. Possible ways to alleviate privacy leakage include rate-limiting API access, monitoring the malicious query behavior, and providing limited predictions for each query. Edge-obfuscation strategies like the one proposed by Joshi and Mishra [25] could slow Maui, but would not completely mitigate it given enough queries. In addition to the defense of the system, malicious query detection should be addressed. Web-based attacks, such as code injection attacks, pose a direct threat to the security of account details.

Future works. For Maui, to make it more applicable to larger datasets, we plan to investigate strategies that help filter out redundant node pairs and improve scalability. We leave this as future work. This strategy can further be used on target models that output noisy node embeddings. Based on Maui, we envision that attacks may be built against federated learning systems, where the clients only have access to a subset of node features and graph structure, and a client may steal private data from other clients.

Furthermore, attacks on different target components (e.g., node features) under different settings are worth studying. The attacks during the training stage are not yet well-studied for GNNs. Gradients with sensitive information can cause serious privacy leakage [81, 82], especially in federated learning settings [39]. Finally, we emphasize that the development of effective countermeasures against privacy attacks is essential. Existing defenses are not suitable for certain attacks [8, 76], such as model extraction attacks.

8 CONCLUSION

In this paper, we have presented Maui, the first link inference attack that operates under the most challenging security setting. By leveraging the vertical partition of graph structures and node features, Maui is capable of accurately inferring the existence of edges between nodes using only black-box access to an inference API. Our evaluation on six real-world datasets has demonstrated the effectiveness of Maui in varying scopes, even under defense mechanisms. The results of this study highlight the pressing need for new privacy-preserving techniques to defend against GNN privacy attacks. With only API access to social media GNNs, Maui can learn who your friends are.

ACKNOWLEDGMENTS

The authors thank anonymous reviewers for their constructive feedback. This work was supported in part by National Science Foundation grant 212720.

REFERENCES

[1] Michael Backes, Mathias Humbert, Jun Pang, and Yang Zhang. 2017. Walk2friends: Inferring Social Links from Mobility Profiles. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)*. 1943–1957.

[2] KM Borgwardt, CS Ong, S Schönauer, SV Vishwanathan, AJ Smola, and HP Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics (Oxford, England)* 21 (2005), i47–56.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and deep locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR 2014*.

[4] Chaochao Chen, Jun Zhou, Longfei Zheng, Huiwen Wu, Lingjuan Lyu, Jia Wu, Bingzhe Wu, Ziqi Liu, Li Wang, and Xiaolin Zheng. 2022. Vertically Federated Graph Neural Network for Privacy-Preserving Node Classification. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. 1959–1965.

[5] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *International Conference on Learning Representations*.

[6] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. 2021. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 896–911.

[7] Tsz-Him Cheung, Weihang Dai, and Shuhan Li. 2021. Fedsgc: Federated simple graph convolution for node classification. In *IJCAI Workshops*.

[8] Enyan Dai, Tianxiang Zhao, Huaisheng Zhu, Junjie Xu, Zhimeng Guo, Hui Liu, Jiliang Tang, and Suhang Wang. 2022. A Comprehensive Survey on Trustworthy Graph Neural Networks: Privacy, Robustness, Fairness, and Explainability. *arXiv preprint arXiv:2204.08570* (2022).

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 3844–3852.

[10] Austin Derrow-Pinion, Jennifer She, David Wong, Oliver Lange, Todd Hester, Luis Perez, Marc Nunkesser, Seongjae Lee, Xueying Guo, Brett Wiltshire, Peter W. Battaglia, Vishal Gupta, Ang Li, Zhongwen Xu, Alvaro Sanchez-Gonzalez, Yujia Li, and Petar Velickovic. 2021. ETA Prediction with Graph Neural Networks in Google Maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM '21)*. 3767–3776.

[11] Haodong Duan, Yue Zhao, Kai Chen, Dahua Lin, and Bo Dai. 2022. Revisiting skeleton-based action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2969–2978.

[12] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. 2020. Quantifying privacy leakage in graph embedding. In *MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. 76–85.

[13] Cynthia Dwork. 2006. Differential Privacy. In *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006) (Lecture Notes in Computer Science, Vol. 4052)*. 1–12.

[14] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. 2019. A Fair Comparison of Graph Neural Networks for Graph Classification. In *International Conference on Learning Representations*.

[15] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[16] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*. 1322–1333.

[17] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Combining Neural Networks with Personalized PageRank for Classification on Graphs. In *International Conference on Learning Representations*.

[18] Neil Zhenqiang Gong and Bin Liu. 2016. You Are Who You Know and How You Behave: Attribute Inference Attacks via Users' Social Friends and Behaviors. In *25th USENIX Security Symposium (USENIX Security 16)*. 979–995.

[19] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks*, Vol. 2. 729–734.

[20] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., 1025–1035.

[21] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. 2021. Stealing Links from Graph Neural Networks. In *USENIX Security Symposium (USENIX Security)*.

[22] Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. 2021. Node-level membership inference attacks against graph neural networks. *arXiv preprint arXiv:2102.05429* (2021).

[23] I-Chung Hsieh and Cheng-Te Li. 2021. NetFense: Adversarial Defenses against Privacy Attacks on Neural Networks for Graph Data. *IEEE Transactions on Knowledge and Data Engineering* (2021), 1–1.

[24] Vassilis N. Ioannidis, Xiang Song, Saurav Manchanda, Mufei Li, Xiaoqin Pan, Da Zheng, Xia Ning, Xiangxiang Zeng, and George Karypis. 2020. DRKG - Drug Repurposing Knowledge Graph for Covid-19. https://github.com/gnn4dr/DRKG/.

[25] Rucha Bhalchandra Joshi and Subhankar Mishra. 2022. Edge-level privacy in Graph Neural Networks. In *18th International Workshop on Mining and Learning with Graphs*.

[26] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2014. Private analysis of graph structure. *ACM Transactions on Database Systems (TODS)* 39, 3 (2014), 1–33.

[27] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. 2019. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11–20.

[28] Isaiah J King, Xiaokui Shu, Jiyong Jang, Kevin Eykholt, Taesung Lee, and H Howie Huang. 2023. EdgeTorrent: Real-time Temporal Graph Representations for Intrusion Detection. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*. 77–91.

[29] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[30] Harlin Lee, Andrea L Bertozzi, Jelena Kovačević, and Yuejie Chi. 2022. Privacy-Preserving Federated Multi-Task Linear Regression: A One-Shot Linear Mixing Approach Inspired By Graph Regularization. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5947–5951.

[31] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. 2018. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing* 67, 1 (2018), 97–109.

[32] Kaiyang Li, Guangchun Luo, Yang Ye, Wei Li, Shihao Ji, and Zhipeng Cai. 2020. Adversarial privacy-preserving graph embedding against inference attack. *IEEE Internet of Things Journal* 8, 8 (2020), 6904–6915.

[33] Kailai Li, Jiawei Sun, Ruoxin Chen, Wei Ding, Kexue Yu, Jie Li, and Chentao Wu. 2023. Towards Practical Edge Inference Attacks Against Graph Neural Networks. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1–5.

[34] Mengcheng Li, Liang An, Hongwen Zhang, Lianpeng Wu, Feng Chen, Tao Yu, and Yebin Liu. 2022. Interacting attention graph for single image two-hand reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2761–2770.

[35] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards Deeper Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. Association for Computing Machinery, 338–348.

[36] Yugeng Liu, Rui Wen, Xinlei He, Ahmed Salem, Zhikun Zhang, Michael Backes, Emiliano De Cristofaro, Mario Fritz, and Yang Zhang. 2022. ML-Doctor: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models. In *31st USENIX Security Symposium (USENIX Security 22)*.

[37] Zhiwei Liu, Liangwei Yang, Ziwei Fan, Hao Peng, and Philip S Yu. 2022. Federated social recommendation with graph neural network. *ACM Transactions on Intelligent Systems and Technology (TIST)* 13, 4 (2022), 1–24.

[38] Xinjian Luo, Yuncheng Wu, Xiaokui Xiao, and Beng Chin Ooi. 2021. Feature Inference Attack on Model Predictions in Vertical Federated Learning. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 181–192.

[39] Lingjuan Lyu, Han Yu, and Qiang Yang. 2020. Threats to federated learning: A survey. *arXiv preprint arXiv:2003.02133* (2020).

[40] Guangxu Mei, Ziyu Guo, Shijun Liu, and Li Pan. 2019. Sgnn: A graph neural network based federated learning approach by hiding structure. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2560–2568.

[41] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5115–5124.

[42] Jacob Levy Moreno. 1934. *Who shall survive? A new approach to the problem of human interrelations*. Nervous and Mental Disease Publishing, Chapter VIII. Community Organization.

[43] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*. arXiv:2007.08663

[44] Iyiola E Olatunji, Wolfgang Nejdl, and Megha Khosla. 2021. Membership inference attack on graph neural networks. In *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 11–20.

[45] Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of*

the Association for Computational Linguistics 5 (2017), 101–115.
[46] Daniele Romanini, Adam James Hall, Pavlos Papadopoulos, Tom Titcombe, Abbas Ismail, Tudor Cebere, Robin Sandmann, Robin Roehm, and Michael A Hoeh. 2021. Pyvertical: A vertical federated learning framework for multi-headed splitnn. *arXiv preprint arXiv:2104.00489* (2021).
[47] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. 2021. Multi-scale attributed node embedding. *Journal of Complex Networks* 9, 2 (2021), cnab014.
[48] Benedek Rozemberczki and Rik Sarkar. 2020. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM '20)*. 1325–1334.
[49] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*. 593–607.
[50] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of Graph Neural Network Evaluation. *Relational Representation Learning Workshop, NeurIPS 2018* (2018).
[51] Yun Shen, Xinlei He, Yufei Han, and Yang Zhang. 2022. Model Stealing Attacks Against Inductive Graph Neural Networks. In *2022 IEEE Symposium on Security and Privacy (S&P)*. 1175–1192.
[52] Martin Simonovsky and Nikos Komodakis. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*. 412–422.
[53] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. 2020. Graph clustering with graph neural networks. *arXiv preprint arXiv:2006.16904* (2020).
[54] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018).
[55] Binghui Wang, Jiayi Guo, Ang Li, Yiran Chen, and Hai Li. 2021. Privacy-preserving representation learning on graphs: A mutual information perspective. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1667–1676.
[56] Yu Wang and Lichao Sun. 2021. Membership inference attacks on knowledge graphs. *arXiv preprint arXiv:2104.08273* (2021).
[57] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. 2019. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591* (2019).
[58] Colby Wise, Vassilis N. Ioannidis, Miguel Romero Calvo, Xiang Song, George Price, Ninad Kulkani, Ryan Brand, Parminder Bhatia, and George Karypis. 2020. COVID-19 knowledge graph: Accelerating information retrieval and discovery for scientific literature. In *AACL-IJCNLP 2020 Workshop on Integrating Structured Knowledge and Neural Networks for NLP (KNLP)*.
[59] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. 2021. Adapting membership inference attacks to gnn for graph classification: Approaches and implications. In *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1421–1426.
[60] Bang Wu, Xiangwen Yang, Shirui Pan, and Xingliang Yuan. 2022. Model Extraction Attacks on Graph Neural Networks: Taxonomy and Realisation. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (ASIA CCS '22)*. 337–350.
[61] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925* (2021).
[62] F. Wu, Y. Long, C. Zhang, and B. Li. 2022. LinkTeller: Recovering Private Edges from Graph Neural Networks via Influence Analysis. In *2022 2022 IEEE Symposium on Security and Privacy (S&P)*. 522–541.
[63] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. 2020. Graph information bottleneck. *Advances in Neural Information Processing Systems* 33 (2020), 20437–20448.
[64] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (2021), 4–24.
[65] Wensheng Xia, Ying Li, Lan Zhang, Zhonghai Wu, and Xiaoyong Yuan. 2021. A vertical federated learning framework for horizontally partitioned labels. *arXiv preprint arXiv:2106.10056* (2021).
[66] Yiqing Xie, Sha Li, Carl Yang, Raymond Chi-Wing Wong, and Jiawei Han. 2020. When Do GNNs Work: Understanding and Improving Neighborhood Aggregation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. 1303–1309.
[67] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*.
[68] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, James Joshi, and Heiko Ludwig. 2021. Fedv: Privacy-preserving federated learning over vertically partitioned data. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*. 181–192.
[69] Yang Xu, Xuexian Hu, Jianghong Wei, Hongjian Yang, and Kejia Li. 2023. VF-CART: A communication-efficient vertical federated framework for the CART algorithm. *Journal of King Saud University-Computer and Information Sciences* 35, 1 (2023), 237–249.
[70] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of The 33rd International Conference on Machine Learning*, Vol. 48. 40–48.
[71] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 7370–7377.
[72] Haina Ye, Xinzhou Cheng, Mingqiang Yuan, Lexi Xu, Jie Gao, and Chen Cheng. 2016. A survey of security and privacy in big data. In *2016 16th international symposium on communications and information technologies (iscit)*. IEEE, 268–272.
[73] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.
[74] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. 4805–4815.
[75] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. In *Advances in Neural Information Processing Systems*, Vol. 32.
[76] He Zhang, Bang Wu, Xingliang Yuan, Shirui Pan, Hanghang Tong, and Jian Pei. 2022. Trustworthy Graph Neural Networks: Aspects, Methods and Trends. *arXiv e-prints* (May 2022), arXiv:2205.07424.
[77] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. 5171–5181.
[78] Yi Zhang, Yuying Zhao, Zhaoqing Li, Xueqi Cheng, Yu Wang, Olivera Kotevska, Philip S Yu, and Tyler Derr. 2023. A Survey on Privacy in Graph Neural Networks: Attacks, Preservation, and Applications. *arXiv preprint arXiv:2308.16375* (2023).
[79] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. 2022. Inference Attacks Against Graph Neural Networks. In *31st USENIX Security Symposium (USENIX Security 22)*. 4543–4560.
[80] Zaixi Zhang, Qi Liu, Zhenya Huang, Hao Wang, Chengqiang Lu, Chuanren Liu, and Enhong Chen. 2021. GraphMI: Extracting Private Graph Data from Graph Neural Networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. 3749–3755.
[81] Junyi Zhu and Matthew B Blaschko. 2020. R-GAP: Recursive Gradient Attack on Privacy. In *International Conference on Learning Representations*.
[82] Ligeng Zhu, Zhijian Liu, and Song Han. 2019. Deep leakage from gradients. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 14774–14784.

A DETAILS OF MODEL PERFORMANCE

Table 12 shows the model performance on the six datasets. Specifically, GCN-2l$_{(\cdot)}$ is a 2-layer GCN model with a defense mechanism as the countermeasure to the attacks, which will be elaborated in Appendix C.

Table 12: Accuracy (in percent) of GNN predictions.

Model	Cora	Citeseer	Coauthor-CS	Facebook	Github	LastFM-Asia
GCN-2l	84.3	73.0	92.6	91.1	86.5	80.5
GAT-2l	86.7	73.4	92.5	91.5	86.7	80.4
GCN-2l$_{(LG)}$	82.1	71.0	90.7	85.8	84.7	77.6
GCN-2l$_{(EA)}$	86.0	76.0	91.4	88.4	86.4	80.1

B PERFORMANCE WITH 3-LAYER GCNS

Due to enormous time consumption of Maui on specific graphs, we instead present the results of Maui with alternative estimation strategies. We report the attacks with 3-layer GCNs by AP and the running time. Here, we do not consider 1-layer GNNs or deeper GNNs. We can identify all the correct node pairs with 1-layer GNNs simply by finding the influence nodes. For deeper GNNs, the node

Table 13: Performance with 3-layer GCNs.

Type	Attack	Cora	Citeseer	Coauthor-CS	Facebook	Github	LastFM-Asia
Local	Decoder	28.9	54.0	61.2±2.6	51.1±1.9	46.3±4.6	30.8±1.1
	LSA2-post	27.8	51.5	59.4±1.8	49.3±1.2	20.5±5.4	32.3±1.5
	LSA2-feat	32.8	56.5	83.3±1.3	38.8±1.3	21.0±4.8	32.5±0.9
	LinkTeller	77.7	85.6	85.5±0.5	79.5±1.0	81.8±1.9	82.4±0.6
	MAUI$_{(E)}$	91.7	92.5	87.3±0.4	89.4±0.9	85.1±3.3	94.0±0.1
	MAUI$_{(C)}$	**93.3**	**97.0**	**96.0±0.2**	**96.0±0.4**	**92.0±2.2**	**96.6±0.1**
Global	Decoder	11.8	24.6	34.6±3.4	24.2±2.0	21.4±6.8	13.6±0.9
	LSA2-post	8.1	20.3	32.5±1.8	17.4±0.8	2.9±1.0	10.6±1.0
	LSA2-feat	12.4	27.6	65.7±2.7	14.6±0.8	4.2±1.6	12.1±0.6
	LinkTeller	64.1	74.4	71.6±1.0	62.2±0.8	59.9±4.0	69.4±1.3
	MAUI$_{(E)}$	86.6	85.4	75.6±0.7	82.9±0.8	60.0±10.1	91.4±0.1
	MAUI$_{(C)}$	**88.5**	**92.4**	**90.3±0.4**	**89.9±0.6**	**75.5±5.2**	**94.6±0.1**
Model Accuracy		*83.1*	*69.8*	*92.1*	*93.7*	*86.8*	*82.2*

representations face over-smoothing problems [35], leading to the performance degradation of both GNNs and attacks.

Attack performance. From Table 13, MAUI$_{(C)}$ obtains the best results. Compared with the results with 2-layer GNNs in Table 4, the performance of LSA2s declines largely due to the huge increase of influence nodes. The performance of LinkTeller drops slightly, same as the finding from Wu et al. [62]. In the global attack, MAUI$_{(E)}$ obtains 10.3% more, and the performance of MAUI$_{(C)}$ has improved by 7.0% in comparison with Table 5.

Running time. In Table 14, LinkTeller maintains the best results with 3-layer GCNs. Overall, MAUIs require huge time consumption. The results of MAUI$_{(C)}{}^{+}$ in Table 14 indicate the running time for the original estimation strategy, so the total running time of MAUI$_{(C)}$ is the sum of MAUI$_{(E)}$ and MAUI$_{(C)}{}^{+}$. Compared with Table 8, the extra time consumption of LinkTeller with 3-layer GCNs is from the increasing nonzero influence values, while the time complexity stays constant. Differently, the time complexity of MAUIs rises largely due to the increase of GNN layers, which leads to considerable running time in practice. The other factor associated with the time complexity of MAUIs is the graph density. As mentioned previously, Github graph is uneven and contains subgraphs with large density.

Table 14: Running time (in seconds) with 3-layer GCNs.

Attack	Cora	Citeseer	Coauthor-CS	Facebook	Github	LastFM-Asia
LSA2-post	24.4	19.2	12.3±0.3	18.4±0.6	21.1±10.1	20.0±0.8
LSA2-feat	**21.9**	18.2	11.7±0.2	17.0±0.4	18.4±8.3	18.0±0.7
LinkTeller	28.3	**15.6**	**7.4±1.0**	**9.8±0.7**	**7.3±2.3**	**16.8±0.8**
MAUI$_{(E)}$	517.9	179.4	61.4±10.4	120.5±5.4	179.0±189.5	278.2±23.5
MAUI$_{(C)}{}^{+}$	155.9	141.6	64.8±5.6	125.6±13.7	994.6±1222.4	192.1±16.1
$\mathcal{I}$	*27.7*	*52.6*	*32.9±0.1*	*23.2±0.7*	*33.2±34.1*	*27.3±0.6*

C DETAILS OF DEFENSE MECHANISM

LapGraph. Given a value of ϵ and a randomization generator, LapGraph takes a graph as input, then generates a perturbed graph that guarantees ϵ-edge DP. In edge DP [26], two undirected graphs are considered neighbors if one graph can be obtained from the other by adding or removing one edge. Thereby, edge DP defense mechanisms operate directly on adjacency matrix. LapGraph applies a specific level of Laplace noise to each cell of the adjacency matrix, based on the privacy budget. The intuition of adjacency matrix perturbation is to guarantee the indistinguishability between two neighboring adjacency matrices by adding enough noise. To further enhance the discreteness of the adjacency matrix, LapGraph retains only the top-T cells from the perturbed adjacency matrix and converts it into a sparse binary matrix. Here, T represents a random number approximated to the number of edges in the original graph. This discrete transformation maintains sparsity, mitigating excessive memory consumption while ensuring the compatibility with spatial GNNs. To better control the amount of perturbation, LapGraph adopts ϵ-DP mechanism [13, 62]. It controls the edge density via a small portion of the privacy budget ϵ and applies the remaining budget to generate the Laplace noise. The larger ϵ is, the less perturbation will be conducted. According to the analysis from Wu et al. [62], we set ϵ to 10 in our experiment to make sure the original graph is mostly retained and the model performance does not drop significantly.

Table 15: Performance with 2-layer GCNs under LapGraph, with the perturbed graphs as ground truth.

Type	Attack	Cora	Citeseer	Coauthor-CS	Facebook	Github	LastFM-Asia
Local	Decoder	48.2	63.5	76.8±1.6	63.5±0.5	61.6±2.8	48.9±0.7
	LSA2-post	41.8	61.1	71.8±1.1	60.5±0.4	41.8±6.0	50.1±1.3
	LSA2-feat	42.5	59.2	75.3±0.6	43.9±0.3	32.0±5.6	43.5±0.8
	LinkTeller	75.5	84.7	88.5±0.8	84.4±0.6	88.3±1.8	81.0±0.4
	MAUI	**95.2**	**96.3**	**99.5±0.1**	**97.1±0.3**	**95.9±1.0**	**96.3±0.4**
Global	Decoder	28.1	38.6	55.0±2.6	36.5±1.1	29.8±3.0	26.9±0.5
	LSA2-post	19.0	33.8	51.3±1.5	30.9±1.2	10.2±4.7	25.6±1.3
	LSA2-feat	22.1	36.1	64.5±1.2	24.2±0.9	11.0±4.3	23.6±0.7
	LinkTeller	60.8	72.6	76.6±1.6	69.0±1.3	69.0±6.9	66.2±1.0
	MAUI	**90.5**	**89.1**	**97.4±0.4**	**88.5±0.6**	**85.9±1.9**	**89.9±1.0**

The results in Table 15 are calculated with the perturbed graphs as the ground truth. In particular, the results of MAUI are all above 95.0% in the local attack and 85.0% in the global attack. With such information, a surrogate GNN model can be constructed.

EdgeAttn. EdgeAttn is inspired by the concept of Graph Information Bottleneck (GIB) [63]. The model aims to predict with the minimum sufficient graph structure information. We assign imbalanced attentions to different edges in each layer of GNNs to ensure the model focuses on the most important graph structure so as to reduce the accuracy of influence estimation. EdgeAttn can be considered as GCNs with edge attention manipulator or adapted 1-head GATs, which do not perturb the original graph structure. The attention for each edge at the l-th layer is calculated as:

$$\alpha_{i,j}^{l} = \exp\left([\boldsymbol{h}_i^{l-1}W^l \Big\| \boldsymbol{h}_j^{l-1}W^l]\boldsymbol{a}^l\right) \Big/ \sum_{v_j \in \mathcal{N}_i \cup \{v_i\}} \exp\left([\boldsymbol{h}_i^{l-1}W^l \Big\| \boldsymbol{h}_j^{l-1}W^l]\boldsymbol{a}^l\right). \tag{5}$$

Different from GATs, we remove the LeakyReLU function to create imbalanced edge attentions to better protect the edge information. Then, we apply noises to the edge attentions:

$$\hat{\alpha}_{i,j}^{l} = \alpha_{i,j}^{l} + o, \tag{6}$$

where o is the noise from a uniform distribution $o \sim U(-\mu, \mu)$. μ is a hyperparameter, which we set to 0.05 in the experiment. The noise is

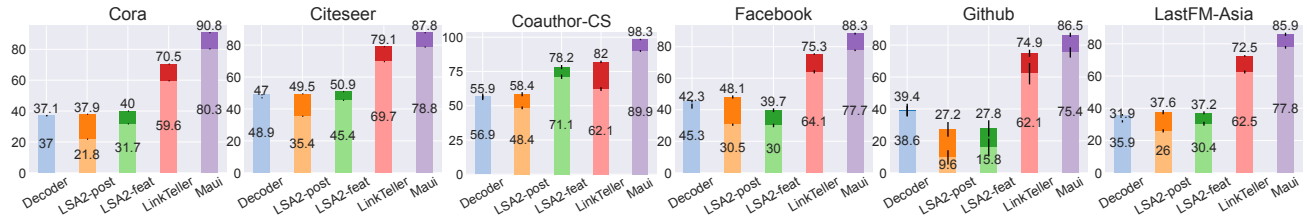


Figure 5: Precision of the global attack w/ and w/o the knowledge of node degree. The precision is calculated with top-100% identified edges. The light and the dark bars represent the result w/ and w/o the extra knowledge, respectively.

Algorithm 4: Global Attack with Node Degrees

Input: The normalized influence values $\tilde{\Theta}$; node degrees $deg(V)$
Output: The inferred adjacency matrix $\hat{A}$

Set an empty adjacency matrix $\hat{A}$ for V
foreach $v_i \in V$ **do**
  Recover the top-$(deg(v_i))$ edges in $\hat{A}$
$\hat{A} \leftarrow \hat{A} | \hat{A}^T$
$\tilde{\Theta} \leftarrow \tilde{\Theta} + \tilde{\Theta}^T$
foreach $v_i \in V$ **do**
  Sort $\tilde{\boldsymbol{\theta}}_i$ in ascending order
  foreach $(v_i, v_j) \in \hat{A}$ **do**
    if $|\hat{A}_i| > deg(v_i)$ ***and*** $|\hat{A}_j| > deg(v_j)$ **then**
      Remove (v_i, v_j) and (v_j, v_i) in $\hat{A}$
Recover the top-$(deg(V))$ edges in $\hat{A}$ according to $\tilde{\Theta}$

added to edge attentions from each GNN layer during both training and inference stage. This noise perturbs the edge attention, influencing predictions within an acceptable range. Consequently, the prediction probabilities vary slightly with each inference, thereby protecting the edge information. During the training stage, we introduce element-wise entropy to edge attentions of each layer to encourage the imbalance of edge attentions. The ultimate loss function is set to be:

$$L_{\text{sum}} = L_c(P, Y) + \beta \sum_{1 \le l \le L} L_e\left(\alpha^l\right), \tag{7}$$

where L_c is the cross-entropy, L_e is the element-wise entropy, and β is the hyperparameter to control the imbalance of edge attentions. We follow the same training settings in our experiment and dynamically choose the value of β to ensure the model performance under EdgeAttn is slightly better than LapGraph.

D ANALYTICAL STUDIES

D.1 Global Attack with Node Degrees

Here, we consider the node degree as extra knowledge. We present the global attack strategy in Algorithm 4, when we know the degree of each node. We treat the edges as directional. The first step is to find the incoming edges of each node individually, according to the influence values of node pairs and the node degrees. Then we convert the edges into non-directional edges. The second step is to trim the redundant edges and make sure the top-k edges are identified.

We report the global attack with the knowledge of node degrees in Figure 5. The result is the best result the attacks can achieve in the global attack, as the global incompatibility is mostly eliminated. The visible dark bars in Figure 5 are the improvement for the incompatibility, averaging 8.6% for all and 11.2% for black-box attacks. We conclude that if an attack performs well in the local attack, with some extra knowledge, the whole graph will be in serious danger of privacy leakage.

D.2 Attack with Original Node Features

We report the results of Maui when using the original node features in Table 16. The performance declines compared with using the random initialized features. We set the same features for each node, which alleviates the unequal prediction contribution from each node, so that the influence scores largely depend on the graph structure, specifically the distance between two nodes. From using Proposition 1, we ensure that closer nodes have more influence on each others' predictions.

Table 16: Performance of Maui when using the original node features.

Dataset	GCNs		GATs	
	local	global	local	global
Cora	81.8	69.0	85.5	78.5
Citeseer	89.4	78.5	91.8	83.7
Coauthor-CS	89.7±0.4	80.1±1.0	96.0±0.3	90.8±0.8
Facebook	78.7±0.6	61.0±0.8	85.4±0.9	74.1±1.7
Github	84.0±1.6	62.8±5.2	91.7±0.5	82.2±2.3
LastFM-Asia	80.5±0.7	61.8±1.4	86.1±0.5	79.3±1.0