GraphMI: Extracting Private Graph Data from Graph Neural Networks

Zaixi Zhang[1] , **Qi Liu**[1*] , **Zhenya Huang**[1] , **Hao Wang**[1] , **Chengqiang Lu**[2] , **Chuanren Liu**[3] , **Enhong Chen**[1]
[1]Anhui Province Key Lab. of Big Data Analysis and Application,
School of Computer Science and Technology, University of Science and Technology of China
[2]Alibaba Group
[3]The University of Tennessee Knoxville
{zaixi, huangzhy, wanghao3}@mail.ustc.edu.cn, {qiliuql, cheneh}@ustc.edu.cn,
lulu.lcq@alibaba-inc.com, cliu89@utk.edu

Abstract

As machine learning becomes more widely used for critical applications, the need to study its implications in privacy turns to be urgent. Given access to the target model and auxiliary information, model inversion attack aims to infer sensitive features of the training dataset, which leads to great privacy concerns. Despite its success in grid-like domains, directly applying model inversion techniques on non-grid domains such as graph achieves poor attack performance due to the difficulty to fully exploit the intrinsic properties of graphs and attributes of nodes used in Graph Neural Networks (GNN). To bridge this gap, we present **Graph M**odel **I**nversion attack (GraphMI), which aims to extract private graph data of the training graph by inverting GNN, one of the state-of-the-art graph analysis tools. Specifically, we firstly propose a projected gradient module to tackle the discreteness of graph edges while preserving sparsity and smoothness of graph features. Then we design a graph auto-encoder module to efficiently exploit graph topology, node attributes, and target model parameters for edge inference. With the proposed methods, we study the connection between model inversion risk and edge influence and show that edges with greater influence are more likely to be recovered. Extensive experiments over several public datasets demonstrate the effectiveness of our method. We also show that differential privacy in its canonical form can hardly defend our attack while preserving decent utility.

1 Introduction

Machine learning (ML) algorithms based on deep neural networks have achieved remarkable success in a range of domains such as computer vision [Zhang *et al.*, 2020], natural language processing [Liu *et al.*, 2020], and graph analysis [Wang *et al.*, 2019]. Meanwhile, the impact of machine learning techniques on privacy is receiving more and more attention because many machine learning applications involve processing sensitive user data (e.g., purchase records) [Fioretto *et al.*, 2020]. Attackers may exploit the output (i.e., black-box attack) or the parameters (i.e., white-box attack) of machine learning models to potentially reveal sensitive information in training data.

According to the attacker's goal, privacy attacks can be categorized into several types, such as membership inference attack [Shokri *et al.*, 2017], model extraction attack [Tramèr *et al.*, 2016], and model inversion attack [Fredrikson *et al.*, 2015]. Of particular interest to this paper is model inversion attack which aims to extract sensitive features of training data given output labels and partial knowledge of non-sensitive features. Model inversion attack was firstly introduced by [Fredrikson *et al.*, 2014], where an attacker, given a linear regression model for personalized medicine and some demographic information about a patient, could predict the patient's genetic markers. Generally, model inversion relies on the correlation between features and output labels and try to maximize a posteriori (MAP) or likelihood estimation (MLE) to recover sensitive features. Recently, efforts have been made to extend model inversion to attack other machine learning models, in particular Convolutional Neural Networks (CNN) [Fredrikson *et al.*, 2015; Aïvodji *et al.*, 2019]. Thus far, most model inversion attacks are investigated in the grid-like domain (e.g., images), leaving its effect on the non-grid domain (e.g., graph structured data) an open problem.

Graph Neural Network (GNN) as one of the state-of-the-art graph analysis tools shows excellent results in various applications on graph-structured data [Kipf and Welling, 2017; Veličković *et al.*, 2018]. However, the fact that many GNN-based applications such as recommendation systems [Wu *et al.*, 2019b] and social relationship analysis [Wang *et al.*, 2019] rely on processing sensitive graph data raises great privacy concerns. Studying model inversion attack on GNNs helps us understand the vulnerability of GNN models and enable us to avoid privacy risks in advance.

Motivation scenario: Figure 1 shows one concrete motivation scenario of model inversion attack on GNN models. Users' friendships are sensitive relational data, and users

*Contact Author

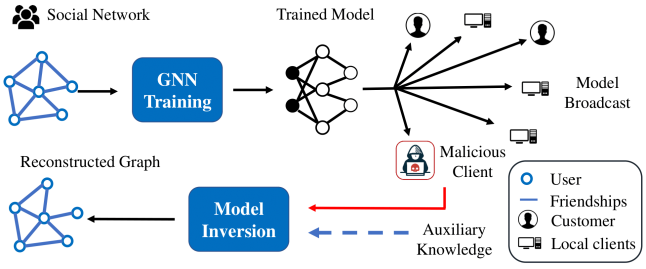


Figure 1: One motivation scenario in social networks

want to keep them private. Sometimes social network data is collected with user permission to train GNN models for better service. For example, these trained GNN models are used to classify friends or recommend advertisements. Then, the trained models are broadcast to customers or local clients. If the attacker can obtain the trained GNN model from malicious clients, with some auxiliary knowledge crawled from the internet, model inversion attack can be performed to reconstruct friendships among users.

In this paper, we draw attention to model inversion attacks extracting private graph data from GNN models. We focus on the following adversarial scenario. Given the trained GNN model and some auxiliary knowledge (node labels and attributes), the adversary aims to reconstruct all the edges among nodes in the training dataset. However, model inversion attack on graphs brings unique challenges. Firstly, existing model inversion attack methods can barely be applied to the graph setting due to the discrete nature of graphs. Different from the continuous image data, gradient computation and optimization on binary edges of the graph are difficult. Secondly, current model inversion methods fail to exploit the intrinsic properties of graph such as sparsity and feature smoothness. In addition, existing model inversion attack methods cannot fully leverage the information of node attributes and GNN models. For example, node pairs with similar attributes or embeddings are more likely to have edges.

To address the aforementioned challenges, we propose **Graph Model Inversion** attack (GraphMI) for edge reconstruction. GraphMI is designed with two important modules: the projected gradient module and the graph auto-encoder module. The projected gradient module is able to tackle the edge discreteness via convex relaxation while preserving graph sparsity and feature smoothness. The graph auto-encoder module is designed to take all the information of node attributes, graph topology and target model parameters into consideration for graph reconstruction. Based on GraphMI, we investigate the relation between edge influence and model inversion risk and find that edges with greater influence are more likely to be reconstructed. Furthermore, we show that differential privacy, in its canonical form, is of little avail to defend against GraphMI. Experimental results on several public datasets show the effectiveness of GraphMI [1].

[1]https://github.com/zaixizhang/GraphMI

2 Related Work

Based on the attacker's goal, privacy attacks can be categorized into several types such as membership inference attack [Shokri *et al.*, 2017], model extraction attack [Tramèr *et al.*, 2016] and model inversion attack [Fredrikson *et al.*, 2015]. Membership inference attack tries to determine whether one sample was used to train the machine learning model; Model extraction attack is one black-box privacy attack. It tries to extract information of model parameters and reconstruct one substitute model that behaves similarly to the target model. Model inversion attack, which is the focus of this paper, aims to reconstruct sensitive features corresponding to labels of target machine learning models.

Model inversion attack was firstly presented in [Fredrikson *et al.*, 2014] for linear regression models. [Fredrikson *et al.*, 2015] extended model inversion attack to extract faces from shallow neural networks. They cast the model inversion as an optimization problem and solve the problem by gradient descent with modifications to the images. Furthermore, several model inversion attacks in the black-box setting or assisted with generative methods are proposed [Aïvodji *et al.*, 2019; Zhang *et al.*, 2020] in the image domain. Thus far, no existing model inversion attack has focused on the graph domain.

3 Problem Formulation

3.1 Preliminaries on GNNs

One task that GNN models are commonly used for is semi-supervised node classification [Kipf and Welling, 2017]. Given a single network topology with node attributes and a known subset of node labels, GNNs are efficient to infer the classes of unlabeled nodes. Before defining GNN, we firstly introduce the following notations of graph. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected and unweighted graph, where $\mathcal{V}$ is the vertex (i.e. node) set with size $|\mathcal{V}| = N$, and $\mathcal{E}$ is the edge set. Denote $\mathcal{A} \in \{0,1\}^{N\times N}$ as an adjacent matrix containing information of network topology and $X \in \mathbb{R}^{N\times l}$ as a feature matrix with dimension l. In a GNN model, each node i is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^l$ and a scalar label y_i. GNN is used to predict the classes of unlabeled nodes under the adjacency matrix A and the labeled node data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{train}}$. GNN uses all nodes' input features but only $N_{train} < N$ labeled nodes in the training phase.

Formally, the k-th layer of a GNN model obeys the message passing rule and can be modeled by one message passing phase and one readout update phase:

$$m_v^{k+1} = \sum_{u\in\mathcal{N}(v)} \mathbf{M}_k(h_v^k, h_u^k, e_{uv}), \tag{1}$$

$$h_v^{k+1} = \mathbf{U}_k(h_v^k, m_v^{k+1}), \tag{2}$$

where $\mathbf{M}_k$ denotes the message passing function and $\mathbf{U}_k$ is the vertex update function. $\mathcal{N}(v)$ is the neighbors of v in graph $\mathcal{G}$. h_v^k is the feature vector of node v at layer k and e_{uv} denotes the edge feature. $h_v^0 = \mathbf{x}_v$ is the input feature vector of node v.

Specifically, Graph Convolutional Network (GCN) [Kipf and Welling, 2017], a well-established method for semi-

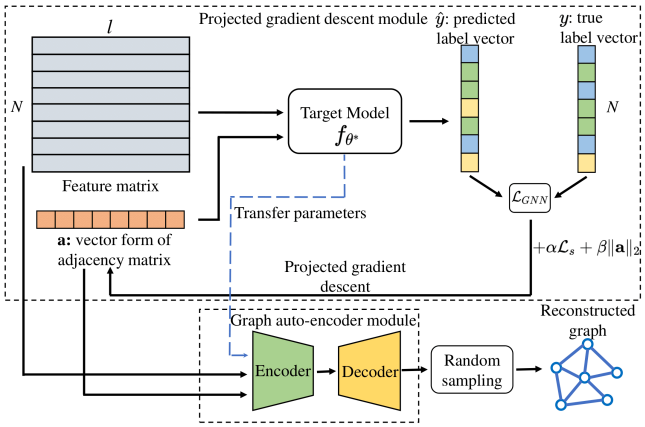


Figure 2: Overview of GraphMI

supervised node classification, obeys the following rule to aggregate neighboring features:

$$H^{k+1} = \sigma\big(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^k W^k\big), \tag{3}$$

where $\hat{A} = A + I_N$ is the adjacency matrix of the graph $\mathcal{G}$ with self connections added and $\hat{D}$ is a diagonal matrix with $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$. $\sigma(\cdot)$ is the ReLU function. H^k and W^k are the feature matrix and the trainable weight matrix of the k-th layer respectively. $H^0 = X$ is the input feature matrix. Note that in most of this paper, we focus on two-layer GCN for the node classification. Later, we show that our graph model inversion attack can be also performed on other types of GNNs, including GAT [Veličković *et al.*, 2018] and GraphSAGE [Hamilton *et al.*, 2017].

3.2 Problem Definition

We refer to the trained model subjected to model inversion attack as the target model. In this paper, we will firstly train a GNN for node classification task from scratch as the target model. We assume a threat model similar to the existing model inversion attacks [Fredrikson *et al.*, 2015].

Attacker's Knowledge and Capability: We will focus on the white-box setting. The attacker is assumed to have access to the target model f and can employ some inference technique to discover the adjacency matrix A of the training graph. In most of the paper, we assume the attacker has labels of all the nodes. In addition to the target model f and node labels, the attacker may have other auxiliary knowledge to facilitate model inversion such as node attributes, node IDs or edge density. We will discuss the impact of auxiliary knowledge and the number of node labels on attack performance in the following sections.

Model Inversion of Graph Neural Networks: Let θ be the model parameters of target model f. During the training phase, f is trained to minimize the loss function $\mathcal{L}(\theta, X, A, Y)$:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta, X, A, Y), \tag{4}$$

where Y is the vector of node labels and X is the feature matrix. Given the trained model and its parameters, graph model inversion aims to find the adjacency matrix A^* that maximizes the posterior possibility:

$$A^* = \arg\max_{A} P(A|X, Y, \theta^*). \tag{5}$$

Algorithm 1 GraphMI

Input: Target GNN model $f_{\theta*}$; Node label vector Y; Node feature matrix X; Learning rate, η_t; Iterations T;
Output: Reconstructed A

$\mathbf{a}^{(0)}$ is set to zeros
Let $t = 0$
while t <T **do**
  Gradient descent: $\mathbf{a}^{(t)} = \mathbf{a}^{(t-1)} - \eta_t \nabla \mathcal{L}_{attack}(\mathbf{a})$;
  Call Projection operation in (13)
end while
Call Graph auto-encoder module in (14)
Call Random sampling module.
return A

4 Proposed Algorithm

Next we introduce GraphMI, our proposed model inversion attack on GNN models.

4.1 Attack Overview

Figure 2 shows the overview of GraphMI. Generally, GraphMI is one optimization-based attack method, which firstly employs projected gradient descent on the graph to find the "optimal" network topology for node labels. Then the adjacency matrix and feature matrix will be sent to the graph auto-encoder module of which parameters are transfered from the target model. Finally, we can interpret the optimized graph as the edge probability matrix and sample a binary adjacency matrix. We summarize GraphMI in Algo 1.

4.2 Details of Modules

Projected Gradient Descent Module: We treat model inversion on GNNs as one optimization problem: given node features or node IDs, we want to minimize the cross-entropy loss between true labels y_i and predicted labels $\hat{y}_i$ from the target GNN model $f_{\theta*}$. The intuition is that the reconstructed adjacency matrix will be similar to the original adjacency matrix if the loss between true labels and predicted labels is minimized. The attack loss on node i is denoted by $\ell_i(A, f_{\theta^*}, \mathbf{x}_i, y_i)$ where A is the reconstructed adjacency matrix, θ^* is the model parameter of the target model f and $\mathbf{x}_i$ is the node feature vector of node i. The objective function can be formulated as:

$$\min_{A\in\{0,1\}^{N\times N}} \mathcal{L}_{GNN}(A) = \frac{1}{N}\sum_{i=1}^{N} \ell_i(A, f_{\theta^*}, \mathbf{x}_i, y_i) \quad s.t. \quad A = A^\top. \tag{6}$$

In many real-world graphs, such as social networks, citation networks, and web pages, connected nodes are likely to have similar features [Wu *et al.*, 2019a]. Based on this observation, we need to ensure the feature smoothness in the

optimized graph. The feature smoothness can be captured by the following loss term $\mathcal{L}_s$:

$$\mathcal{L}_s = \frac{1}{2}\sum_{i,j=1}^{N} A_{i,j}(\mathbf{x}_i - \mathbf{x}_j)^2, \tag{7}$$

where $A_{i,j}$ indicates the connection between node v_i and v_j in the optimized graph and $(\mathbf{x}_i - \mathbf{x}_j)^2$ measures the feature difference between v_i and v_j. $\mathcal{L}_s$ can also be represented as:

$$\mathcal{L}_s = tr(X^\top L X), \tag{8}$$

where $L = D - A$ is the laplacian matrix of A and D is the diagonal matrix of A. In this paper, to make feature smoothness independent of node degrees, we use the normalized lapacian matrix $\hat{L} = D^{-1/2} L D^{-1/2}$ instead:

$$\mathcal{L}_s = tr(X^\top \hat{L} X) = \frac{1}{2}\sum_{i,j=1}^{N} A_{i,j}(\frac{\mathbf{x}_i}{\sqrt{d_i}} - \frac{\mathbf{x}_j}{\sqrt{d_j}})^2, \tag{9}$$

where d_i and d_j denote the degree of node v_i and v_j. To encourage the sparsity of graph structure, F norm of adjacency matrix A is also added to the loss function. The final objective function is:

$$\begin{aligned} \arg\min_{A\in\{0,1\}^{N\times N}} \mathcal{L}_{attack} &= \mathcal{L}_{GNN} + \alpha\mathcal{L}_s + \beta\|A\|_F \\ s.t. \quad A &= A^\top, \end{aligned} \tag{10}$$

where α and β are hyper-parameters that control the contribution of feature smoothing and graph sparsity. Solving equation (10) is a combinatorial optimization problem due to edge discreteness. For ease of gradient computation and update, we firstly replace the symmetric reconstructed adjacency matrix A with its vector form $\mathbf{a}$ that consists of $n := N(N-1)/2$ unique variables in A. Adjacency matrix A and vector $\mathbf{a}$ can be converted to each other easily, which ensures the optimized adjacency matrix is symmetric. Then we relax $\mathbf{a} \in \{0,1\}^n$ into convex space $\mathbf{a} \in [0,1]^n$. We can perform model inversion attack by firstly solving the following optimization problem:

$$\arg\min_{\mathbf{a}\in[0,1]^n} \mathcal{L}_{attack} = \mathcal{L}_{GNN} + \alpha\mathcal{L}_s + \beta\|\mathbf{a}\|_2. \tag{11}$$

The continuous optimization problem 11 is solved by projected gradient descent (PGD):

$$\mathbf{a}^{t+1} = P_{[0,1]}\big[\mathbf{a}^t - \eta_t g_t\big], \tag{12}$$

where t is the iteration index of PGD, η_t is the learning rate, g_t is the gradients of loss $\mathcal{L}_{attack}$ in 10 evaluated at $\mathbf{a}^t$, and

$$P_{[0,1]}[x] = \begin{cases} 0 & x < 0 \\ 1 & x > 1 \\ x & otherwise \end{cases} \tag{13}$$

is the projection operator.

Graph Auto-encoder Module: In GraphMI, we propose to use graph auto-encoder (GAE) [Kipf and Welling, 2016] to post-process the optimized adjacency matrix A. GAE is composed of two components: encoder and decoder. We transfer part of the parameters from the target model f_{θ^*} to the encoder. Specifically, feature matrix and adjacency vector $\mathbf{a}$ are sent to the f_{θ^*} and the node embedding matrix Z is generated by taking the penultimate layer of the target model f_{θ^*}, which is denoted as $H_{\theta^*}(\mathbf{a}, X)$. Then the decoder will reconstruct adjacency matrix A by applying logistic sigmoid function to the inner product of Z:

$$A = \text{sigmoid}(ZZ^\top), \text{with } Z = H_{\theta^*}(\mathbf{a}, X). \tag{14}$$

The node embeddings generated by the graph auto-encoder module encode the information from node attributes, graph topology, and the target GNN model. Intuitively, node pairs with close embeddings are more likely to form edges.

Random Sampling Module: After the optimization problem is solved, the solution A can be interpreted as a probabilistic matrix, which represents the possibility of each edge. We could use random sampling to recover the binary adjacency matrix; see details in the appendix.

4.3 Analysis on Correlation between Edge Influence and Inversion Risk

In previous work [Wu *et al.*, 2016], researchers found feature influence to be an essential factor in incurring privacy risk. In our context of graph model inversion attack, sensitive features are edges. Here we want to characterize the correlation between edge influence and inversion risk. Given label vector Y, adjacency matrix A and feature matrix X, the performance of target model f_{θ^*} for the prediction can be measured by prediction accuracy:

$$ACC(f_{\theta^*}, A, X) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(f^i_{\theta^*}(A, X) = y_i), \tag{15}$$

where, $f^i_{\theta^*}(A, X)$ is the predicted label for node i. The influence of edge e can be defined as:

$$\mathcal{I}(e) = ACC(f_{\theta^*}, A, X) - ACC(f_{\theta^*}, A_{-e}, X), \tag{16}$$

where A_{-e} denotes removing the edge e from the adjacency matrix A. [Wu *et al.*, 2016] proposed to use adversary advantage to characterize model inversion risk of features. The model inversion advantage of adversary $\mathcal{A}$ is defined to be $P[\mathcal{A}(X, f_{\theta^*}) = e] - 1/2$, where $P[\mathcal{A}(X, f_{\theta^*}) = e]$ is the probability that adversary $\mathcal{A}$ correctly infer the existence of edge e. Next, we introduce our theorem.

Theorem 1. *The adversary advantage is greater for edges with greater influence.*

We defer the proof to the appendix. Intuitively, edges with greater influence are more likely to be recovered by GraphMI because these edges have a greater correlation with the model output. In the following section, we will validate our theorem with experiments.

5 Experiments

In this section, we present the experimental results to show the effectiveness of GraphMI. Specifically, our experiments are designed to answer the following research questions:

Method	Cora		Citeseer		Polblogs		USA		Brazil		AIDS		ENZYMES	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP	AUC	AP
Attr. Sim.	0.803	0.808	**0.889**	**0.891**	-	-	-	-	-	-	0.731	0.727	0.564	0.567
MAP	0.747	0.708	0.693	0.755	0.688	0.751	0.594	0.601	0.638	0.661	0.642	0.653	0.617	0.643
GraphMI	**0.868**	**0.883**	0.878	0.885	**0.793**	**0.797**	**0.806**	**0.813**	**0.866**	**0.888**	**0.802**	**0.809**	**0.678**	**0.684**

Table 1: Results of model inversion attack on Graph Neural Networks

- **RQ1**: How effective is GraphMI?
- **RQ2**: Which edges are more likely to be
- **RQ3**: Is differential privacy an effective countermeasure against model inversion attacks on GNN?

5.1 Experimental Settings

Datasets: Our graph model inversion attack method is evaluated on 7 public datasets from 4 categories. The detailed statistics of them are listed in the appendix.

- *Citation Networks*: We use Cora and Citeseer [Sen *et al.*, 2008]. Here, nodes are documents with corresponding bag-of-words features and edges denote citations among nodes. Class labels denote the subfield of research that the papers belong to.
- *Social Networks*: Polblogs [Adamic and Glance, 2005] is the network of political blogs whose nodes do not have features.
- *Air-Traffic Networks*: The air-traffic networks are based on flight records from USA and Brazil. Each node is an airport and an edge indicates a commercial airline route between airports. Labels denote the level of activity in terms of people and flights passing through an airport [Ribeiro *et al.*, 2017].
- *Chemical Networks*: AIDS [Riesen and Bunke, 2008] and ENZYMES [Borgwardt *et al.*, 2005] are chemical datasets that contain many molecure graphs, each node is an atom and each link represents chemical bonds.

Target Models: In our evaluation, we use 3 state-of-the-art GNN models: GCN [Kipf and Welling, 2017], GAT [Veličković *et al.*, 2018] and GraphSAGE [Hamilton *et al.*, 2017]. The parameters of the models are the same as those set in the original papers. To train a target model, 10% randomly sampled nodes are used as the training set. All GNN models are trained for 200 epochs with an early stopping strategy based on convergence behavior and accuracy on a validation set containing 20% randomly sampled nodes. In GraphMI attack experiments, attackers have labels of all the nodes and feature vectors. All the experiments are conducted on Tesla V100 GPUs.

Parameter Settings: In experiments, we set $\alpha = 0.001$, $\beta = 0.0001$, $\eta_t = 0.1$ and $T = 100$ as the default setting. We show how to find optimal values for hyper-parameters in the following section.

Metrics: Since our attack is unsupervised, the attacker cannot find a threshold to make a concrete prediction through the algorithm. To evaluate our attack, we use AUC (area under the ROC curve) and AP (average precision) as our metrics, which is consistent with previous works [Kipf and Welling, 2016]. In experiments, we use all the edges from the training graph and the same number of randomly sampled pairs of unconnected nodes (non-edges) to evaluate AUC and AP.

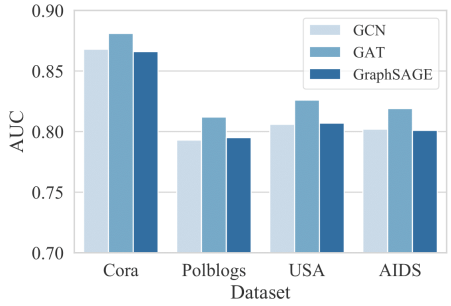


Figure 3: Attack performance of GraphMI on different Graph Neural Networks.

5.2 Results and Discussions

Attack Performance. Results for model inversion attacks on GCN are summarized in table 1. There are two baseline methods, attribute similarity (abbreviated as Attr. Sim.) and MAP. Attribute similarity is measured by cosine distance among node attributes, which is commonly used in previous works [He *et al.*, 2021]. We adapt the model inversion method from [Fredrikson *et al.*, 2015], MAP to the graph neural network setting as the other baseline. Note that some datasets such as Polblogs dataset do not have node attributes, so that we assign one-hot vectors as their attributes. They are not applicable for the attack based on attribute similarity. As can be observed in table 1, GraphMI achieves the best performance across nearly all the datasets, which demonstrates the effectiveness of GraphMI. One exception is Citesser where the attack performance of GraphMI is relatively lower than attribute similarity, which could be explained by more abundant node attribute information of Citeseer compared with other datasets. Thus using node attribute similarity alone could achieve good performance in the Citesser dataset.

In figure 3, we show the attack performance of GraphMI on three GNNs. We observe that GraphMI has better attack performance on GAT model. This may be explained by the fact that GAT model is more powerful and is able to build a stronger correlation between graph topology and node labels. GraphMI can take advantage of such a stronger correlation and achieve better attack performance.

In figure 4, we present the influence of node label propor-

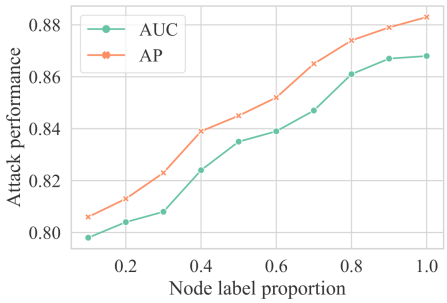
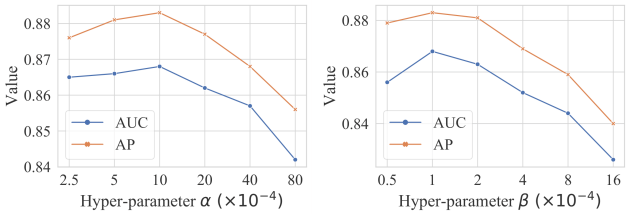


Figure 4: Impact of node label proportion.



(a) (b)

Figure 5: Results of parameter analysis on Cora dataset

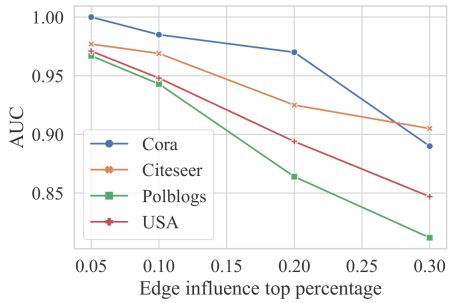


Figure 6: Impact of edge influence on the performance of the GraphMI attack.

Method	ACC	GraphMI AUC
$\epsilon = 1.0$	0.48	0.60
$\epsilon = 5.0$	0.65	0.72
$\epsilon = 10.0$	0.78	0.84
no DP	0.80	0.87

Table 2: The performance of the GraphMI attack against GCN trained with differential privacy on Cora dataset

tion on attack performance. As can be observed from the plot, with fewer node labels, the attack performance will drop gradually. But GraphMI can still achieve over 80 % AUC and AP when only 20% node labels are available, which again verifies the effectiveness of GraphMI.

We also explore the sensitivity of hyper-parameters α and β for GraphMI. In the experiments, we alter the value of α and β to see how they affect the performance of GraphMI. Specifically, we vary α from 0.00025 to 0.008 and β from 0.00005 to 0.0016 in a log scale of base 2. The attack performance change of GraphMI is illustrated in Fig 5. As we can observe, the attack performance of GraphMI can be boosted when choosing proper values for all the hyper-parameters.

Edge Influence. We do experiments to verify our claim that edges with greater influence are more likely to be inferred successfully through model inversion attack. Note that it will be very time-consuming to measure the influence of each edge exactly. According to equation (16), removing edges with greater influence will cause greater drop of prediction accuracy. To select edges with great influence, we apply the state-of-the-art topology attack [Xu *et al.*, 2019] on graphs by removing edges. In Figure 6, we show that for edges with top 5% influence GraphMI achieves the attack AUC of nearly 1.00 in Cora dataset. This implies that the privacy leakage will be more severe if sensitive edges are those with greater influence.

Defense Performance of Differentail Privacy. Differential privacy (DP) is one general approach for protecting privacy. Here, we investigate the impact of differential privacy on GraphMI attacks. (ϵ, δ) - DP is ensured by adding Gaussian noise to clipped gradients in each training iteration [Abadi *et al.*, 2016]. In experiments, δ is set to 10^{-5} and the noise scale is varied to obtain target GNN models with different ϵ from 1.0 to 10.0. The GraphMI attack performance and their model utility are presented in Table 2. As the privacy budget ϵ drops, the performance of GraphMI attack deteriorates at the price of a huge utility drop. Generally, enforcing DP on target models cannot prevent GraphMI attack.

6 Conclusion

In this paper, we presented GraphMI, a model inversion attack method against Graph Neural Networks. Our method was specifically designed and optimized for extracting private graph-structured data from GNNs. Extensive experimental results showed its effectiveness on several state-of-the-art graph neural networks. We also explored and evaluated the impact of node label proportion and edge influence on the attack performance. Finally, we showed that imposing differential privacy on graph neural networks can hardly protect privacy while preserving decent utility.

This paper provided potential tools for investigating the privacy risks of deep learning models on graph-structured data. Interesting future directions include: 1) Extending the current work to a black-box setting.2) Design countermeasures with a better trade-off between utility and privacy.

Acknowledgments

This research was partially supported by grants from the National Natural Science Foundation of China (Grants No.61922073 and U20A20229), and the Fundamental Research Funds for the Central Universities (Grant No.WK2150110021).

References

[Abadi *et al.*, 2016] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS 2016*, pages 308–318, 2016.

[Adamic and Glance, 2005] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, 2005.

[Aïvodji *et al.*, 2019] Ulrich Aïvodji, Sébastien Gambs, and Timon Ther. Gamin: An adversarial approach to black-box model inversion. *arXiv preprint arXiv:1909.11835*, 2019.

[Borgwardt *et al.*, 2005] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

[Fioretto *et al.*, 2020] Ferdinando Fioretto, Lesia Mitridati, and Pascal Van Hentenryck. Differential privacy for stackelberg games. In *IJCAI-20*, pages 3480–3486, 7 2020. Main track.

[Fredrikson *et al.*, 2014] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *23rd {USENIX} Security Symposium*, pages 17–32, 2014.

[Fredrikson *et al.*, 2015] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *SIGSAC*, pages 1322–1333, 2015.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, pages 1024–1034, 2017.

[He *et al.*, 2021] Xinlei He, Jin-Yuan Jia, M. Backes, N. Gong, and Y. Zhang. Stealing links from graph neural networks. In *30th USENIX Security Symposium*, Vancouver, B.C., August 2021. USENIX Association.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *Bayesian Deep Learning Workshop (NeurIPS 2016)*, 2016.

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

[Liu *et al.*, 2020] Xin Liu, Kai Liu, Xiang Li, Jinsong Su, Yubin Ge, Bin Wang, and Jiebo Luo. An iterative multi-source mutual knowledge transfer framework for machine reading comprehension. In *IJCAI-20*, pages 3794–3800, 7 2020.

[Ribeiro *et al.*, 2017] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *SIGKDD*, pages 385–394, 2017.

[Riesen and Bunke, 2008] Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *Joint IAPR International Workshops on SPR and SSPR*, pages 287–297. Springer, 2008.

[Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[Shokri *et al.*, 2017] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[Tramèr *et al.*, 2016] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium*, pages 601–618, 2016.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *ICLR 2018*, 2018.

[Wang *et al.*, 2019] Hao Wang, Tong Xu, Qi Liu, Defu Lian, Enhong Chen, Dongfang Du, Han Wu, and Wen Su. Mcne: An end-to-end framework for learning multiple conditional network representations of social network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1064–1072, 2019.

[Wu *et al.*, 2016] Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F Naughton. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th CSF*, pages 355–370. IEEE, 2016.

[Wu *et al.*, 2019a] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv preprint arXiv:1903.01610*, 2019.

[Wu *et al.*, 2019b] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *AAAI*, volume 33, pages 346–353, 2019.

[Xu *et al.*, 2019] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. *IJCAI*, 2019.

[Zhang *et al.*, 2020] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: generative model-inversion attacks against deep neural networks. In *CVPR*, pages 253–261, 2020.