Label-Only Membership Inference Attack against \\Node-Level Graph Neural Networks

Conti, M.; Li, Jiaxin; Picek, S.; Xu, J.

DOI
10.1145/3560830.3563734

Publication date
2022

Document Version
Final published version

Published in
AISec 2022 - Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security, co-located with CCS 2022

Label-Only Membership Inference Attack against Node-Level Graph Neural Networks

Mauro Conti
University of Padua & Delft University of Technology
Padova, Italy
conti@math.unipd.it

Jiaxin Li*
University of Padua
Padova, Italy
jiaxin.li@studenti.unipd.it

Stjepan Picek
Radboud University & Delft University of Technology
Nijmegen, Netherlands
stjepan.picek@ru.nl

Jing Xu
Delft University of Technology
Delft, Netherlands
j.xu-8@tudelft.nl

ABSTRACT

Graph Neural Networks (GNNs), inspired by Convolutional Neural Networks (CNNs), aggregate the message of nodes' neighbors and structure information to acquire expressive representations of nodes for node classification, graph classification, and link prediction. Previous studies have indicated that node-level GNNs are vulnerable to Membership Inference Attacks (MIAs), which infer whether a node is in the training data of GNNs and leak the node's private information, like the patient's disease history. The implementation of previous MIAs takes advantage of the models' probability output, which is infeasible if GNNs only provide the prediction label (label-only) for the input.

In this paper, we propose a label-only MIA against GNNs for node classification with the help of GNNs' flexible prediction mechanism, e.g., obtaining the prediction label of one node even when neighbors' information is unavailable. Our attacking method achieves around 60% accuracy, precision, and Area Under the Curve (AUC) for most datasets and GNN models, some of which are competitive or even better than state-of-the-art probability-based MIAs implemented under our environment and settings. Additionally, we analyze the influence of the sampling method, model selection approach, and overfitting level on the attack performance of our label-only MIA. All of those three factors have an impact on the attack performance. Then, we consider scenarios where assumptions about the adversary's additional dataset (shadow dataset) and extra information about the target model are relaxed. Even in those scenarios, our label-only MIA achieves a better attack performance in most cases. Finally, we explore the effectiveness of possible defenses, including Dropout, Regularization, Normalization, and Jumping knowledge. None of those four defenses prevent our attack completely.

*corresponding author.

AISec '22, November 11, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9880-0/22/11…$15.00
https://doi.org/10.1145/3560830.3563734

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Machine learning**;

KEYWORDS

Machine learning, Membership inference attack, Graph neural networks

ACM Reference Format:
Mauro Conti, Jiaxin Li, Stjepan Picek, and Jing Xu. 2022. Label-Only Membership Inference Attack against Node-Level Graph Neural Networks. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security (AISec '22), November 11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3560830.3563734

1 INTRODUCTION

Graph Neural Networks (GNNs) are gaining more and more attention for their broad application in analyzing social networks, recommender systems, and biological networks. Node classification, which predicts the label for nodes in the graph, is one of the popular tasks [7, 11, 27]. During the training of GNNs, the model recursively updates nodes' representation from neighbors' message and structure information, similar to the convolution operation of Convolutional Neural Networks (CNNs) on an image. Due to this message passing way, GNNs achieve a competitive performance on multiple tasks [21, 31]. Still, GNNs are vulnerable to Membership Inference Attacks (MIAs), which infer the existence of nodes in the training data of GNNs. In other words, the adversary can determine which nodes belong to the training data of GNNs with the implementation of MIAs.

Studies about MIAs against GNNs are dedicated to finding, defending, and understanding such attacks. Researches on MIAs are critical since they can cause privacy leakage. Let us consider the example of a GNN whose training data contains the social network and symptoms of patients infected with COVID-19 for investigating [17], recognizing, and classifying infection factors. Attacking this GNN, the adversary can utilize its probability vector output to predict whether one patient is in the training data or not. If the adversary can implement MIA with only the prediction label (label-only) of GNN, it will target more models and lead to more severe damage. The adversary knows which persons are in the training data of GNN after attacking with varying MIAs. In the mentioned target GNN, being in the training data means the person is a patient.

Therefore, the adversary can obtain a list of infected patients via MIAs. There is no doubt that the history of the disease is private information for a patient. Therefore, studying varying MIAs against GNNs is essential and significant.

There are two types of MIAs: one is classifier-based MIA, and the other is non-classifier-based MIA. The classifier-based MIA leverages a binary classifier (called attack model), trained with attack features, to infer whether or not one data point is a member of training data. In this paper, the attack features are properties of one data point (or node) for feeding into the attack model. The non-classifier-based MIA directly or indirectly calculates a metric value for one data point and compares this value with a threshold for determining membership. The original intuition of MIA is that the overfitting model will assign a higher probability value to the training data than the testing data. Hence, previous MIAs against GNNs utilize the prediction probability vector to act as attack features or compute metric values. However, they are ineffective when the model only outputs the input's label, which is a label-only condition. Furthermore, previous label-only MIAs against CNNs and semantic segmentation models are not effective or challenging to implement while being transferred to GNNs. To improve the attack performance under the label-only condition, we propose our label-only MIA against GNNs, which is more efficient and straightforward than previous methods.

The critical point of our label-only MIA is the attack features extraction for training the attack model. Under the label-only condition, the adversary cannot obtain the prediction probability vector from the target model for the attack feature extraction. Usually, the adversary has access to a shadow dataset from the same distribution as the target dataset, which is used for training and testing the target model. Therefore, the attacker can train a shadow model with the shadow dataset to mimic the behavior of the target model. To improve the effect of imitation, the adversary could relabel the shadow dataset with the target model and train a relabelled shadow model. Indeed, one previous label-only work (not on GNNs) acquires the prediction probability vector of the relabelled shadow model as the attack features of the target dataset, called the transfer attack [14]. Besides, Li et al. [14] and Choquette-Choo et al. [4] proposed to use the distance between the original data point with its closest adversarial example to implement label-only MIA. However, finding the adversarial example of a graph is difficult by changing the connection and nodes' features under the label-only condition. Therefore, we put forward our label-only MIA.

Unlike previous methods, we observe a flexible prediction mechanism of GNNs, which means that we can obtain the prediction label of a specific node with or without its neighbors' features and connection information fed. Besides, inspired by previous data augmentation methods for obtaining features [4] or reconstructing the prediction probability vector [18], we utilize feature masking and step-by-step edge dropping to measure the stability of the prediction label. Specifically, we obtain the attack features of the attack model from three aspects. The first one is the fixed properties of the node, including the number of neighbors and the ground truth of the node. The second one is the prediction correctness of the target model while only feeding the node's features into the GNN and masking the node's properties with different rates and values. The last one is putting the node's features, features of 1-hop neighbors, and connection information into the GNN. Then, the edges between the node and 1-hop neighbors are dropped step by step, and the node's features are masked with different rates and values to get the prediction correctness of the node and its 1-hop neighbors. We obtain the attack features from those three aspects for training our label-only attack model.

Furthermore, we relax the assumptions that the shadow dataset is from the same distribution as the target dataset and that the GNN architecture and type of the shadow model are the same as the target model. Besides, we explore the influence of the sampling ways, model selection strategies, and the overfitting level on the performance of the attack model. Finally, we explore the robustness of our label-only MIA against several possible defenses.

Our main contributions are:

(1) We propose a label-only MIA, which achieves competitive or better performance than state-of-the-art probability-based MIAs.
(2) We relax the assumptions about the shadow dataset and target model (the GNN architecture and type), which achieves a better performance in most cases.
(3) We explore the influence of the sampling ways, model selection strategies, and the overfitting level on the attack performance. Both of them have an impact on the attack performance.
(4) We analyze the possible defenses against our label-only MIA, where we find that none of the defenses can prevent our attack completely.

Our code is available at https://github.com/fight-think/Label-Only_MIA_for_GNNs.git.

2 BACKGROUND

We introduce some background knowledge about GNN and MIA for further explanation. In Section 2.1, we first review the general GNN architectures. Then, we present the implementation of MIAs in Section 2.2.

2.1 Graph Neural Networks

GNNs utilize the structure information and nodes' features to update nodes' representations for prediction. In this section, we interpret the basic notions and general architectures of GNNs.

2.1.1 Notations. A graph $G = (V, E)$ consists of $|V|$ vertexes and $|E|$ edges. V denotes the set of vertexes $\{V_1, V_2, \ldots, V_{|V|}\}$ with features X=$\{X_1, X_2, \ldots, X_{|V|}\}$. E represents the set of edges $\{E_1, E_2, \ldots, E_{|E|}\}$, each of which connects two nodes in the graph. For node classification tasks, each node V_i has a label Y_i. The purpose of node classification tasks is to predict the label of the node with GNNs. GNNs leverage the node's feature and the message passed from the node's neighborhoods to decide its label on the node classification task. We define the l-hop neighbors of node V_i as $N^l(V_i)$, which contains a set of nodes whose distance with node V_i is equal to or less than l except node V_i itself. For convenience, $N(V_i)$ denotes 1-hop neighbors of node V_i. Correspondingly, node V_i, its l-hop neighbors, connected edges, and relative features make the l-hop subgraph, denoted as $g^l(V_i)$, of node V_i.

2.1.2 General GNN Architecture. The successful CNN applications empirically prove that the local convolution operation can capture an efficient feature representation of an image for downstream classification tasks. Taking advantage of a similar idea, GNNs aggregate and transfer the message of neighbors into the node's representation, which also compresses the graph structure information. We select four commonly used GNNs, including Graph Convolutional Network (GCN) [13], Graph Attention Network (GAT) [26], SAmple and aggreGatE (GraphSAGE) [8], and Graph Isomorphism Network (GIN) [28]. The *AGGREGATION* and *TRANSFORMATION* methods make them distinctive. For each node V_i, $X_i^{(l)}$ is the representation at layer l within the iterative convolution, $H_i^{(l)}$ is its hidden state before the *TRANSFORMATION* operation. $X_i^{(0)}$ is the original feature of node V_i. Thus, the general procedures of convolution are defined in the following equations.

$$H_i^{(l)} = AGGREGATION(\{X_i^{(l-1)}, X_j^{(l-1)}, j \in \{N(V_i)\}\}). \quad (1)$$

$$X_i^{(l)} = TRANSFORMATION(H_i^{(l)}). \quad (2)$$

Here, $N(V_i)$ is the 1-hop neighbors of node V_i. The *AGGREGATION* operation clusters the representations of the current node with its neighbors to obtain its presentation. The *TRANSFORMATION* operation is usually a nonlinear conversion. The distinctions of four GNNs are based on those two operations.

2.2 Membership Inference Attack

The MIA aims to distinguish the training (members) and testing data (non-members) of the target model. Under the context of the node classification, the MIA determines which nodes are in the training data. Generally, there are two types of attack strategies. One is to train a classifier, with which we can predict the possibility of being a member or non-member for a specific data point [20, 22]. The other is to directly or indirectly compute the metric about the data point [4, 14, 15]. Then, the attacker determines the data point as a member by comparing the metric value with a threshold. The label-only MIA in our paper belongs to the first type. Here, we provide more details about the classifier-based MIA.

Classifier-based MIA. The key of classifier-based MIA is acquiring the dataset (also called attack dataset) for training the classifier to distinguish the training and testing data. Usually, the adversary has access to a shadow dataset with the same distribution as the target dataset. With this shadow dataset, the adversary trains a shadow model to mimic the behavior of the target model. Even though the adversary can only query the target model, the adversary has full knowledge of the shadow model. Then, the adversary utilizes the prediction of the shadow model and data split of the shadow model's training and testing data to construct an attack dataset. Finally, the attacker will train an attack model based on the attack dataset for attacking the target model, i.e., distinguishing the training and testing data of the target model.

3 OUR LABEL-ONLY MIA

GNNs for node classification tasks have two categories, inductive and transductive. Under the inductive setting, the GNN cannot access the testing nodes during training. In the transductive environment, the training set consists of training nodes, unlabelled testing nodes, and nodes' connections. In this paper, we do not consider the transductive setting because the testing data partially acts as the training materials and is not separated from the training data, which mismatches with the knowledge of MIA. He et al. [10] also only focused on the inductive setting. While predicting the label of a specific node in the graph, the GNN takes its features, possible neighbors' features, and connection structure information for calculation. However, the GNN could also predict the node's label if only providing the property of a node without neighbors' knowledge. We call this way of prediction the *flexible prediction mechanism*. The flexible way of prediction brings advantages to our label-only MIA, which only gets the label from the model. We formulate our label-only MIA in Section 3.1. Then, we discuss the threat model and attack methodology in Section 3.2 and Section 3.3.

3.1 Problem Formulation

The goal of the MIA is to determine whether the target node V_t belongs to the training data (member) or testing data (non-member) of the target model F_t. To implement the MIA, the adversary has some external knowledge E. We formulate our label-only MIA A as the following function:

$$A : V_t, F_t, E \rightarrow \{1, 0\}. \quad (3)$$

Here, "1" means V_t is in the training data, and "0" otherwise (thus, it is a binary classification task). In the experiments, we train a binary classifier to solve this task.

3.2 Threat Model

The target model F_t is open to the adversary, which means the attacker can query it for the label of the target node V_t. Considering flexible prediction mechanism and practical situation, we limit the query information to the target node's feature, features of its 1-hop neighbors $N(V_t)$, and connection structure $C(V_t)$. There are two reasons for selecting 1-hop neighbors. The first one is that it requires less information than 2-hop neighbors [10] and all the training or testing data [17] while predicting the membership of one node. The second reason is that the information of the target node is finite under the label-only condition. We cannot get enough helpful distinguishable signals without neighbors' information. Therefore, we choose 1-hop neighbors. Besides, we expose the target node's true label Y_{V_t} to the adversary.

The adversary has some external knowledge E. A shadow dataset D_s, extracted from the same distribution as the target dataset D_t used for training and testing the target model F_t, is in the external knowledge. We relax this assumption and sample the shadow dataset from the other distribution in experiments. Furthermore, the adversary knows the target model's architecture, hyperparameters, and training algorithm. With that knowledge, the attacker can train a shadow model F_s to mimic the behavior of the target model. Similarly, we relax the assumption that the adversary knows the GNN architecture and type of the target model within exploration. The relaxation of those two assumptions tests the limits of our label-only MIA under stricter conditions and is closer to the attack

in the real world. He et al. [10] and Olatunji et al. [17] also relaxed those assumptions.

3.3 Attack Methodology

The adversary obtains some external knowledge for the MIA implementation. Then the question is how to apply our label-only MIA on GNNs with that knowledge and still satisfy the threat model. To answer this question, we formulate the final attack model as a binary classifier to predict whether the node is in the training data of the GNN or not. Therefore, the acquisition of the attack features for training the attack model is the crucial point.

3.3.1 General Steps. The overall process is illustrated in Figure 1. First, the total dataset is split into target and shadow datasets. With a shadow dataset and knowledge about the training of the target model, the adversary can train a shadow model to mimic the behavior of the target model. The shadow model and membership situation of nodes in the shadow dataset are transparent to the adversary. Thus, the attacker generates the attack dataset via data points' fixed properties and multiple queries for each data point to the shadow model, which only outputs the prediction label. The features of the data point in the attack dataset are attack features. If the node is in the shadow model's training data, the adversary assigns its attack features with the label "1" and otherwise "0". Then, the adversary trains the attack model with the attack dataset extracted from the shadow dataset and model.

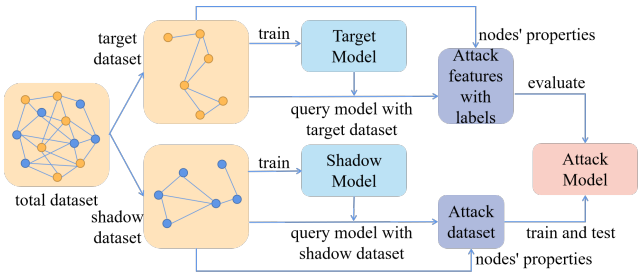

Figure 1: The general process of our label-only MIA.

Similarly, the adversary queries the target model with target nodes from the target dataset to get each target node's attack features with labels, which indicate whether nodes are from the target model's training data ("1") or testing data ("0"). Afterward, the attacker feeds the target nodes' attack features into the attack model to test the attack performance on the target model. Notably, the acquisition of the attack features is interpreted in the next section.

3.3.2 Attack Features Acquisition. Under the label-only condition, the adversary only gets the prediction label of the target node from the target model. Therefore, obtaining the attack features of target nodes in the target dataset is crucial. Previous label-only MIAs [14] (not on GNNs) extract attack features of the target node from the prediction probability vector of relabelled shadow models, which are under the adversary's control. The relabelled shadow model utilizes the target model to relabel the shadow dataset, which is different from the shadow model. One previous method feeds the target node into relabelled shadow models to get the prediction probability vector as the attack features of this node under the label-only condition because the target model does not expose the prediction probability vector. For comparison, we present the result of this transfer attack under the label-only condition in Appendix A.

Unlike previous methods, we construct target nodes' attack features based on fixed properties of the node, data augmentation, flexible prediction mechanism, and practical situation. The application of data augmentation obtains inspiration from previous works [4, 5]. We consider the attack features from three aspects: fixed properties, 0-hop query, and 1-hop query. The specified properties, like the ground truth of the target node in the target model, act as a part of features in previous attack models [12, 22]. Besides, the adversary has only access to 1-hop neighbors, which means the attacker could query the target model with (1-hop query) or without (0-hop query) information of 1-hop neighbors. Therefore, we extract features from the three mentioned aspects.

Acquisition with fixed properties. Olatunji et al. [17] mentioned that the connection between nodes increases the vulnerability of GNN models to privacy attacks. Inspired by this, we count the number of neighbors (n_num) and signal (w_i_node), which indicates whether the node is independent of other nodes, as part of the attack features of the target node V_t. Notably, the neighbors could be in the training or testing data of the dataset while counting the number of neighbors in the target or shadow dataset. Besides, we include the target node's ground truth (o_label) into the attack features, which is the same as the previous MIA [22].

Acquisition with the 0-hop query. The prediction of GNNs is flexible, which means we can get the prediction label of the node with or without the features and the connection information of its neighbors fed into GNNs. For each target node V_t, we randomly change different rates of its feature values X_{V_t} to the largest and smallest value in the feature space. Then we only input this changed feature of the target node to the target model and record whether the prediction is the same as the ground truth of the target node. Here, $i\_none\_max\_rate$ and $i\_none\_min\_rate$ are the record results for each $rate$ in $rate\_set$. The "1" means the prediction of the changed feature is the same as the ground truth of the target node and the "0" otherwise.

Acquisition with the 1-hop query. Due to the flexible prediction mechanism of GNNs, we feed the features X_{V_t} of the target node V_t, the features $X_{N(V_t)}$ of its 1-hop neighbors $N(V_t)$, and connection information $C(V_t)$ between the target node and its 1-hop neighbors into the target model. At the same time, we apply two data augmentation strategies simultaneously. One is randomly changing different rates of feature values to the largest and smallest value in the feature space. The other is dropping the connected edges between the target node and its 1-hop neighbors one by one. Then, we record the prediction accuracy of the 1-hop neighbors and the identity between the prediction label and the target node's ground truth.

Following the previous paragraph, we explain attack features' names while acquiring with the 1-hop query. Figure 2 consists of features with a high absolute SHAP value. The $n\_acc\_all\_max\_rate$ means the accuracy of the neighbors while keeping all the edges and altering a percentage ($rate$) of the target node's feature values to the maximum value in the feature space. The $n\_acc\_none\_max\_rate$ is similar except all the edges are removed. The $n\_acc\_avg\_max\_rate$ is the average accuracy of the neighbors during removing edges.

The *i_all_max_rate* indicates whether the prediction of the target node's changed feature is the same as its ground truth while keeping all the edges. The *i_step_max_rate* presents the rate of cases where the prediction of the target node's changed feature is the same as its ground truth while reducing edges step by step. The names of attack features are slightly different while changing the feature value to the smallest value, i.e., replacing "max" with "min" in the feature name. Besides, we record the real change percentage *change_p_rate* of the feature values while randomly select a percentage (*rate*) of feature values for altering.

Our label-only MIA combines fixed properties, the result of the 0-hop query, and the result of the 1-hop query as the attack features. Previous studies [6, 10, 17] obtained the prediction of the target node with 2-hop neighbors or all nodes in the training or testing data as the input of the target model. They contain more information than our label-only MIA while predicting the node's label. Under the label-only condition, we cannot get the prediction probability vector from the target model. However, while dropping edges or altering its feature values, the target node and its neighbors' prediction correctness situation could also provide the signal for distinguishing the training and testing data.

4 EXPERIMENTS

We first describe datasets in our experiments, followed by describing models' architectures and training settings. In Section 4.3, we introduce evaluation metrics. Finally, we illustrate experimental steps.

4.1 Datasets

We conduct experiments on four datasets: Cora_ML, CiteSeer, DBLP, and PubMed [2]. The main reason for selecting those datasets is that the number of nodes and classes of those datasets varies, which is beneficial for various experiments. Table 1 describes the statistics of those four datasets.

Table 1: Statistical information of datasets.

Dataset	classes	edges	nodes	length of node feature
Cora_ML	7	16,316	2,995	2,879
CiteSeer	6	10,674	4,230	602
DBLP	4	105,734	17,716	1,639
PubMed	3	88,648	19,717	500

4.2 Model Architectures and Training Settings

The GNN types we consider are GCN, GAT, GraphSAGE, and GIN. We vary the model architectures and training settings to explore the influence of the overfitting level on the attack performance. Here, we discuss model architectures and training settings for target models with high and low overfitting levels.

Low overfitting level. In this setting, we set four types of GNNs with three layers (input, hidden, and output layers), with 16 neurons in the hidden layer. The optimization algorithm of GNNs is Adam, with a learning rate of 6e-3 and a weight decay of 0.5. The number of training epochs is 400. Besides, we apply the BatchNorm, Dropout (0.5), and Jumping knowledge (concatenation) [29] to reduce the overfitting level. Jumping knowledge (concatenation) concatenates the output of each layer as the input of the last layer.

High overfitting level. Different from models with low overfitting levels, we set the number of layers to 5 (3 hidden layers) and the number of neurons in the hidden layer to 64. The optimization algorithm of GNNs is Adam, with a learning rate of 0.001 and without weight decay. The number of training epochs is 200. In addition, we do not apply any strategies to reduce the overfitting level because we attempt to get the model with a high overfitting level for comparison.

We decide on the training hyperparameters of models with a high or low overfitting level by increasing or decreasing the overfitting level in a reasonable range. Besides, we refer to the settings in previous works [10, 17].

The attack model is a multilayer perceptron (MLP) with 2 hidden layers, each of which has 64 neurons. The optimization algorithm is Adam, with a learning rate of 0.001. The number of training epochs is 300, and the batch size is 32. We use the Binary Cross Entropy to guide the training of the attack model. Our attack model's architecture is similar to attack models in previous works [10, 17]. After several attempts, we slightly changed the structure and hyperparameters to fit attack features in our label-only MIA. Finally, we obtained the attack model's structure and hyperparameters mentioned before.

4.3 Evaluation Metrics

We train the attack model with the objective function related to the Binary Cross Entropy. The output of the attack model is the probability of the input being a member of training data. We evaluate the attack performance with six metrics: precision, recall, F1 while the threshold is 0.5, accuracy, AUC, and True Positive Rate (TPR) while the False Positive Rate is 0.1, which is inspired by the work of Carlini et al. [3].

4.4 Experimental Steps

The implementation of our label-only MIA is composed of several steps. First, we divide the total dataset into training and testing data for target and shadow models, i.e., four sub-datasets. Then, we generate the attack dataset from the shadow model and shadow dataset for training and testing the attack model. After extracting features and labels from the target model and target dataset, we evaluate the attack performance of the attack model on those features and labels. Apart from implementing our label-only MIA, we also implement previous probability-based MIAs under the same environment and settings. Those settings consist of the same models trained with the same sub-datasets, the training of the attack models, and the model selection strategy. Besides, we explore the impact of several factors on our label-only MIA. Those factors include the overfitting level, sampling methods, and attack model selection strategies. Furthermore, we attempt to relax two assumptions of training the shadow model. Finally, we analyze the effectiveness of some defenses.

Previous Probability-based MIAs. We use eight probability-based MIAs proposed in previous papers [10, 17, 20, 22, 24, 30] to compare our label-only MIA. Those methods can obtain the prediction probability vector from the target model and extract the attack features or metrics from the prediction probability vector, which is

Table 2: The attack performance of our label-only MIA after ten repetitions (low overfitting level).

Dataset	GNN	Used Nodes	Test Acc (target model)	Train Acc (target model)	Our Attack (avg acc, pre, rec, auc, f1, low_fpr_0.01_tpr)
Cora_ML	GAT	1,344	0.736	0.883	[**0.604**, 0.605, 0.618, 0.658, 0.607, 0.059]
	GCN		0.763	0.951	[**0.613**, 0.621, 0.606, 0.666, 0.607, 0.065]
	GIN		0.664	0.875	[**0.59**, 0.624, 0.49, 0.631, 0.536, 0.041]
	GraphSAGE		0.73	0.956	[**0.602**, 0.618, 0.559, 0.65, 0.581, 0.061]
CiteSeer	GAT	3,336	0.793	0.879	[**0.598**, 0.611, 0.585, 0.647, 0.583, 0.032]
	GCN		0.81	0.942	[**0.599**, 0.622, 0.536, 0.645, 0.569, 0.031]
	GIN		0.777	0.921	[**0.592**, 0.605, 0.563, 0.635, 0.571, 0.032]
	GraphSAGE		0.795	0.937	[**0.581**, 0.571, 0.69, 0.617, 0.621, 0.032]
DBLP	GAT	7,920	0.732	0.83	[**0.593**, 0.604, 0.586, 0.645, 0.584, 0.048]
	GCN		0.746	0.884	[**0.593**, 0.598, 0.618, 0.644, 0.596, 0.043]
	GIN		0.715	0.865	[**0.579**, 0.59, 0.563, 0.621, 0.564, 0.026]
	GraphSAGE		0.739	0.891	[**0.579**, 0.578, 0.643, 0.618, 0.595, 0.032]
PubMed	GAT	12,300	0.861	0.884	[**0.583**, 0.573, 0.661, 0.636, 0.61, 0.033]
	GCN		0.867	0.907	[**0.596**, 0.601, 0.58, 0.657, 0.589, 0.059]
	GIN		0.852	0.899	[**0.568**, 0.572, 0.571, 0.611, 0.563, 0.034]
	GraphSAGE		0.866	0.923	[**0.557**, 0.548, 0.665, 0.597, 0.598, 0.033]

impossible under the label-only condition. Four MIAs utilize the classifier to determine membership. They train the attack model based on the attack features. Specifically, those four probability-based MIAs with classifiers use the output of the 0-hop query (top two probability values), the 2-hop query (top two probability values), and the query with feeding training or testing data into the model to predict the label of one node (all probability values). In our paper, we name four probability-based MIAs with classifiers: 0-hop, 2-hop, the combination of 0-hop and 2-hop, and all probability methods. The other four MIAs calculate metrics based on the prediction probability vector. They leverage prediction correctness (Gap Attack), probability of ground truth, cross-entropy, and modified cross-entropy of the prediction to distinguish members and non-members. Notably, those four methods select thresholds with the help of shadow models.

Overfitting. As mentioned in the model architectures and settings, we attempt to explore the influence of the overfitting level on the attack performance. Thus, we train models of different architectures with various settings to expose the impact of the overfitting level on our label-only MIA.

Sampling Methods. We focus on the node-level GNN in this paper. The dataset for training GNNs is composed of a graph with many nodes and edges. As we mentioned, we need to sample four sub-datasets with equal or approximately equal data points in the total dataset. However, the number of nodes in each class is not equal and even has a large gap. Therefore, the sampling cannot guarantee that each class has an equal number of data points in each sub-dataset unless decreasing the number of data points in each sub-dataset. Based on this situation, we take three sampling methods into comparison. The first is a random sampling of four sub-datasets with maximum utilization of data points (called *the random sampling method*). The second one is a strict class-balanced sampling approach (called *the balanced sampling method*), which guarantees non-overlapping, class-balanced, and fewer data points in each sub-dataset. The third one is *the partially balanced sampling method*, which ensures the class balance in the training data of target and shadow models and randomly samples data points for the testing data.

Attack Model Selection Strategies. We train and test the attack model based on the attack dataset. Then, we evaluate the attack performance of the attack model with attack features. The selection of the attack model during the training process impacts the final attack model. Here, we choose the attack model based on five metrics, including training accuracy (*train_acc*), testing accuracy (*test_acc*), training loss (*train_loss*), testing loss (*test_loss*), and evaluation accuracy (*evaluate_acc*). Importantly, the adversary cannot compute evaluation accuracy in a practical scenario because the adversary cannot obtain the attack features of the target dataset for evaluation before selecting the final attack model. We implement the selection strategy based on evaluation accuracy. Finally, we analyze the attack performance of those model selection strategies.

Assumptions Relaxation. The assumptions about the shadow dataset and the target model's information (GNN type and architecture) are not always available in real-world settings. Therefore, we relax those two assumptions alone and together to assess the attack performance of our label-only MIA.

Defenses. The methods for reducing the overfitting level are commonly used to prevent MIAs. To evaluate the effectiveness of our label-only MIA, we apply the Dropout, Regularization, Normalization, and Jumping knowledge to reduce the overfitting level.

5 RESULTS AND DISCUSSIONS

In this section, we provide the results of our experiments and discuss the findings from the results. Section 5.1 compares the attack performance of our label-only MIA with eight previous probability-based MIAs. We interpret the attack model with SHAP values in Section 5.2. In Section 5.3, we explore the influence of different factors on the attack performance of the attack model. We relax two assumptions about the shadow dataset and the target model's information in Section 5.4. Finally, we present the effectiveness of our MIA against four defenses.

5.1 Attack Performance Comparison

As mentioned, the adversary trains the shadow model, which has the same architecture as the target model, with the non-overlapping shadow dataset from the same distribution as the target dataset.

Table 2 provides the attack performance of our label-only MIA after ten repetitions. Repeating each experiment ten times reduces the impact of the randomness, which is the same as the repetition times in the previous work [17].

In this table, the number of data points used in each experiment is related to the balanced sampling method. The overfitting level, measured by the gap between the training and testing accuracy of the target model, is relatively low. From the table, we can see that the attack accuracy, precision, and AUC value are around 0.6 in most experiments, indicating the effectiveness of our label-only MIA against GNNs.

Table 8 in Appendix A shows the corresponding attack performance of four probability-based MIAs with classifiers under the same environment as Table 2. We highlight the average attack accuracy of each table in bold and red. Comparing the average attack accuracy of each row in two tables, we can see that our label-only MIA has higher average accuracy than previous probability-based MIA in most cases. For instance, our label-only MIA achieves the average attack accuracy of 0.596, while four previous probability-based MIAs with classifiers obtain values of 0.506, 0.506, 0.504, and 0.507 of GCN on the PubMed dataset. Table 9 in Appendix A presents the attack performance of the transfer attack and four probability-based MIAs with metrics. Comparing the average attack accuracy (bold and red) of Table 9 and Table 2, we can also find that our label-only MIA has a higher attack performance than those MIAs. It reflects that our label-only MIA has a competitive, even better performance than previous probability-based methods in most experiments under our environment and settings.

5.2 Attack Model Explanation

We describe the details of acquiring the features for training the attack model in Section 3.3.2. To better understand the model's behavior and attempt to explain the model, we calculate the SHAP [16] value of each feature. The SHAP value implies the contribution or importance of each feature to the prediction of the model. Figure 2 gives the SHAP values of each feature in the attack model under the Cora_ML and GCN setting, which has a higher attack performance for clearly exposing the importance of each feature.

The left column in the figure displays features' names, ranked by the absolute SHAP values of each feature over total data points. In the middle, a large number of colorful points represent total data points with different feature values and SHAP values in each row. The x-axis means the SHAP value. The right line implies the feature values for spots in the middle. We can see that the feature named "i_all_min_1.0" has the top absolute SHAP value. This feature presents whether the prediction of the target model to the target node is the same as the ground truth while keeping all the edges between the target node with 1-hop neighbors and changing 100% of the target node's features to the minimum value. In addition, we can observe that most features have an apparent positive or negative impact on model output while being assigned high or low values. Besides, the features with the prefix "n_acc" have higher SHAP values than other features, which exposes that neighboring nodes' accuracy under various settings is beneficial for distinguishing the training and testing data.

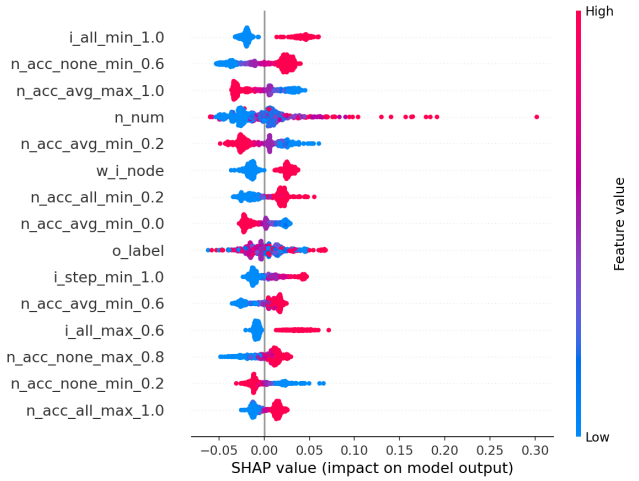

Figure 2: The SHAP values of attack features (top 15). The left column indicates the attack features ranked by the absolute SHAP value. The colorful right line represents the ruler of features' values. In the middle, the cluster of points reflects data points in the dataset. The x-axis shows the SHAP value of each feature within one data point.

5.3 Influence Factors

This section explores our label-only MIA's influence factors, including the overfitting level, sampling methods, and attack model selection strategies.

Overfitting. Table 3 shows the attack performance of our label-only MIA against target models with a high overfitting level. We compare the results in this table with Table 2 and highlight the average attack accuracy in bold and red. The average attack accuracy could increase as the enlargement of the overfitting level—for example, the DBLP on GAT and GCN models, from 0.593 and 0.593 (low overfitting level) to 0.6 and 0.599 (high overfitting level). On the contrary, the increase in the overfitting level could also reduce our label-only MIA's attack performance. For instance, the average attack accuracies move from high (0.604, 0.613, 0.59, and 0.602) to low values (0.597, 0.596, 0.536, and 0.593) under combinations of Cora_ML with GAT, GCN, GIN, and GraphSAGE, respectively. Therefore, the higher overfitting level impacts the attack performance, i.e., it could increase or decrease our label-only MIA's attack accuracy. We use the models with a low overfitting level to complete the subsequent exploration.

Sampling Methods. As mentioned in Section 4.4, we explore the impact of three sampling methods, including the (0) random, (1) balanced, and (2) partially balanced sampling methods. Table 4 exposes the number of data points in four sub-datasets, including the training and testing data of shadow and target models under different sampling methods. In the table, target_train, target_test, shadow_train, and shadow_test represent four sub-datasets. The "total" means the number of data points in this sub-dataset. The "one class" indicates the number of data points in each class of this sub-dataset. The "-" means uncertainty indicating that the number of data points in each class is not fixed due to random selection.

Table 3: The attack performance of our label-only MIA after ten repetitions (high overfitting level).

Dataset	GNN	Test Acc (target model)	Train Acc (target model)	Our Attack (avg acc, pre, rec, auc, f1, low_fpr_0.01_tpr)
Cora_ML	GAT	0.658	0.973	[**0.597**, 0.635, 0.504, 0.652, 0.545, 0.056]
	GCN	0.684	0.96	[**0.596**, 0.605, 0.586, 0.634, 0.586, 0.033]
	GIN	0.298	0.577	[**0.536**, 0.539, 0.525, 0.554, 0.525, 0.022]
	GraphSAGE	0.619	0.999	[**0.593**, 0.602, 0.595, 0.637, 0.59, 0.054]
CiteSeer	GAT	0.748	0.913	[**0.592**, 0.612, 0.546, 0.641, 0.567, 0.042]
	GCN	0.767	0.922	[**0.603**, 0.62, 0.572, 0.656, 0.586, 0.043]
	GIN	0.327	0.578	[**0.538**, 0.536, 0.569, 0.556, 0.549, 0.029]
	GraphSAGE	0.732	0.874	[**0.591**, 0.618, 0.532, 0.637, 0.557, 0.027]
DBLP	GAT	0.716	0.916	[**0.6**, 0.617, 0.585, 0.648, 0.582, 0.044]
	GCN	0.722	0.917	[**0.599**, 0.627, 0.549, 0.653, 0.572, 0.062]
	GIN	0.428	0.652	[**0.555**, 0.561, 0.509, 0.583, 0.522, 0.025]
	GraphSAGE	0.666	0.856	[**0.579**, 0.586, 0.593, 0.618, 0.577, 0.029]
PubMed	GAT	0.828	0.912	[**0.58**, 0.616, 0.477, 0.62, 0.524, 0.033]
	GCN	0.838	0.909	[**0.596**, 0.611, 0.561, 0.653, 0.575, 0.058]
	GIN	0.586	0.639	[**0.522**, 0.52, 0.607, 0.537, 0.542, 0.018]
	GraphSAGE	0.836	0.921	[**0.561**, 0.558, 0.672, 0.594, 0.603, 0.031]

From the table, the uncertainty represents the class imbalance in the random and partially balanced sampling methods.

Table 4: The number of data points in each sub-dataset under different sampling methods.

Dataset	Sampling Method	target_train		target_test		shadow_train		shadow_test	
		total	one class	total	one class	total	one class	total	one class
Cora_ML	0	749	-	749	-	748	-	749	-
	1	336	48	336	48	336	48	336	48
	2	630	90	630	-	630	90	630	-
CiteSeer	0	1,058	-	1,057	-	1,058	-	1,057	-
	1	834	139	834	139	834	139	834	139
	2	600	100	600	-	600	100	600	-
DBLP	0	4,429	-	4,429	-	4,429	-	4,429	-
	1	1,980	495	1,980	495	1,980	495	1,980	495
	2	3,200	800	3,200	-	3,200	800	3,200	-
PubMed	0	4,929	-	4,929	-	4,930	-	4,929	-
	1	3,075	1,025	3,075	1,025	3,075	1,025	3,075	1,025
	2	4,500	1,500	4,500	-	4,500	1,500	4,500	-

Table 5 provides the attack performance of different datasets and GNN models under three sampling methods. From the table, the third sampling method achieves the best average attack accuracy in all datasets and GNN models apart from the GIN model on the CiteSeer dataset, which obtains the highest average attack accuracy with the first sampling method. From Table 4, we can see that both first and third sampling methods suffer class imbalance, which implies class imbalance can improve the attack accuracy. Besides, the difference in average attack accuracy reaches 7% between the second and third sampling methods on the GAT model with the DBLP dataset. It indicates that the sampling method influences the attack performance via class imbalance and achieves a maximum gap of 7% for average attack accuracy. Olatunji et al. [17] used the partially balanced sampling method, while He et al. [10] leveraged the random sampling method. We use the balanced sampling method by default to avoid the disturbance brought by the class imbalance.

Table 5: The average attack accuracy of four datasets and GNN models under three sampling methods after ten repetitions.

Sampling Method	Dataset	GNN	Test Acc (target model)	Train Acc (target model)	Avg Acc
0	Cora_ML	GCN	0.68	0.851	0.621
	CiteSeer	GIN	0.742	0.956	0.613
	DBLP	GAT	0.653	0.682	0.614
	PubMed	GraphSAGE	0.871	0.919	0.55
1	Cora_ML	GCN	0.749	0.935	0.608
	CiteSeer	GIN	0.775	0.927	0.591
	DBLP	GAT	0.732	0.837	0.594
	PubMed	GraphSAGE	0.87	0.925	0.574
2	Cora_ML	GCN	0.777	0.941	0.651
	CiteSeer	GIN	0.728	0.946	0.568
	DBLP	GAT	0.771	0.82	0.664
	PubMed	GraphSAGE	0.866	0.919	0.581

Attack Model Selection Strategies. Section 4.4 explains that we select the attack model during the training process based on five metrics, including training accuracy (*train_acc*), testing accuracy (*test_acc*), training loss (*train_loss*), testing loss (*test_loss*), and evaluation accuracy (*evaluate_acc*). Table 6 gives the average attack accuracy of the selected attack model. From the table, we could observe that selection strategies based on testing accuracy and loss have a slightly better attack accuracy than those based on training accuracy and loss with a maximum gap of 1% average attack accuracy. This paper uses testing accuracy for selecting the attack model.

Under our environment and settings, previous probability-based methods with classifiers might not achieve the attack performance reported in their papers [10, 17]. We attribute this kind of difference to three influence factors analyzed in this section and the change in the attack environment, including the model's architecture, training process, and hyperparameters.

5.4 Assumptions Relaxation

We design and conduct an ablation study on the relaxation of two assumptions about the shadow dataset and the target model's information. In the first experiment, we utilize shadow datasets from other distributions. Secondly, we train the shadow model with different types and architectures from the target model. Finally, we relax those two assumptions together.

Figure 3 shows the average attack accuracy comparison of the first experiment. We fix the target model and shadow model to

Table 6: The average attack accuracy of four datasets and GNN models with five model selection strategies after ten repetitions.

Dataset	GNN	Test Acc (target model)	Train Acc (target model)	Average accuracy of the attack model selected by different strategies				
				train_acc	test_acc	train_loss	test_loss	evaluate_acc
Cora_ML	GCN	0.744	0.943	0.607	0.616	0.603	0.617	0.644
CiteSeer	GIN	0.771	0.924	0.585	0.589	0.582	0.596	0.615
DBLP	GAT	0.734	0.837	0.582	0.592	0.583	0.589	0.604
PubMed	GraphSAGE	0.874	0.924	0.572	0.578	0.565	0.589	0.604

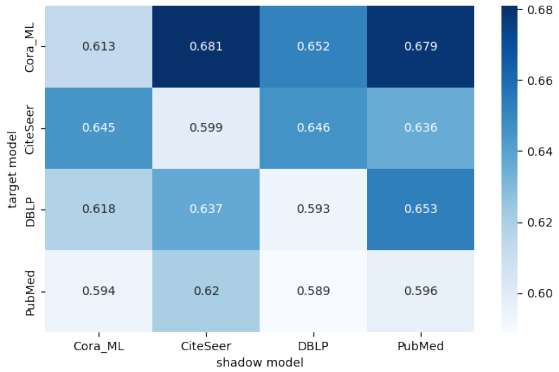

Figure 3: The average attack accuracy after ten repetitions while the shadow dataset is from the other distribution. The fixed GNN for shadow and target models is GCN. The x-axis means the setting of the shadow model. The y-axis represents the setting of the target model.

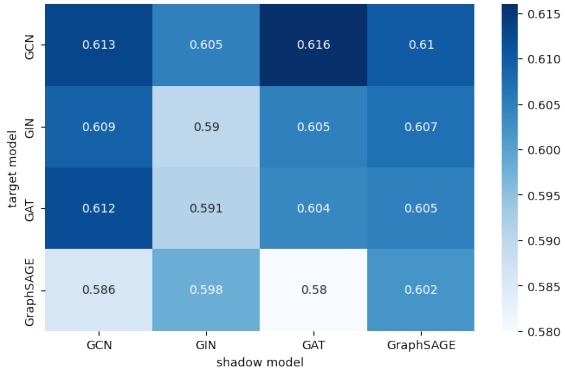

Figure 4: The average attack accuracy after ten repetitions while the target model's information is relaxed. The shadow and target datasets are from Cora_ML.

GCN. While the target models are trained with Cora_ML, CiteSeer, DBLP, and PubMed (each row in the figure), we obtain the largest attack accuracy with shadow datasets from PubMed (0.679), DBLP (0.646), PubMed (0.653), and CiteSeer (0.62), which are not from the same datasets as the target datasets. Figure 4 shows the result of the second experiment. We fix the target dataset and shadow dataset to sample from Cora_ML. The main reason for selecting Cora_ML is that the average attack accuracy of Cora_ML is relatively higher than other datasets, which means the change is evident while relaxing the second assumption. From the figure, we obtain the highest attack accuracy while the target and shadow models are GCN (GAT), GIN (GCN), GAT (GCN), and GraphSAGE (GraphSAGE), most of which do not have the same model type for target and shadow models. Figure 5 illustrates the average attack accuracy of the third experiment. Similarly, the maximum attack accuracy is not from the settings where the target model is trained with the same dataset and model type as the shadow model. From those three figures, we surprisingly find that the relaxation of those two assumptions will increase the attack performance in most cases, which reflects on relatively light colors on the diagonal. Even when the attack performance decreases, the reduction degree is low. Therefore, the attack performance primarily increases while relaxing the assumptions about the shadow dataset and target model's information. The possible reason for this phenomenon is that the attack features of members (or non-members) are similar or are converted to be alike in the attack model under settings where the shadow model's distribution and the target model's information are different. This phenomenon is also discussed in previous works [10, 20].

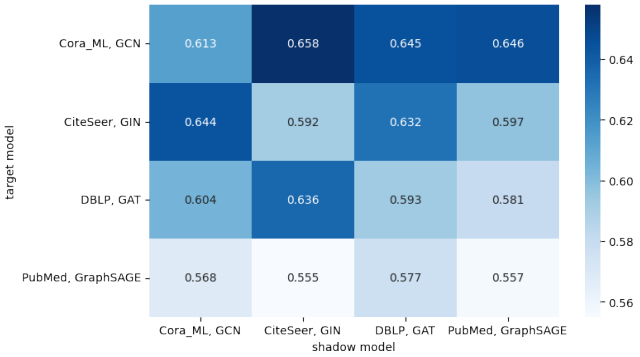

Figure 5: The average attack accuracy after ten repetitions while the shadow dataset and the target model's information are relaxed together.

5.5 Defenses

Table 7 shows the average attack accuracy of ten repetitions against four different defenses. The defenses include Normalization (the BatchNorm), Dropout (0.5), Regularization (the Adam with weight decay of 0.5), and Jumping knowledge (concatenation). Here, we select the GCN model on the Cora_ML dataset with a high overfitting level to investigate the influence of defenses due to its relatively high attack performance. From the table, we can observe that the average accuracy decreases when applying Normalization (row 1, 0.586) or Regularization (row 3, 0.573) compared with no defenses (row 0, 0.596). However, the average accuracy slightly increases when only using the Dropout (row 2, 0.602) or Jumping knowledge (row 4, 0.606). Besides, the combinations between four defenses

could raise (row 5, 7, 9, 11, 12, 13, 15) or lower (row 6, 8, 10, 14) the average accuracy. In addition, Regularization is in four rows (6, 8, 10, 14), where the average accuracy decreases. It does not mean the average accuracy will be lower with Regularization combined with other defenses, like rows 11 and 13. The results show that Regularization and Normalization could slightly decrease the average attack accuracy while applying alone. Moreover, those four defenses cannot prevent our label-only MIA completely with higher average accuracy than 0.5 apart from row 14.

We apply and compare four regular defense mechanisms, all of which cannot prevent our label-only MIA completely. One solution is to utilize a defense mechanism combining GNNs' particular properties. For example, flexible prediction strategy and unique connection between nodes and their neighbors. Currently, we do not find an efficient defense mechanism for our label-only MIA and leave it as future work.

6 RELATED WORK

Membership Inference Attacks. In 2016, Shokri et al. [22] proposed MIAs to machine learning models by training the attack model with attack features through the shadow model and dataset. Then, Salem et al. [20] relaxed some assumptions from [22] and achieved competitive attack performance. Recently, membership inference attacks widely spread to multiple data types and model categories [1, 9, 19, 23, 32]. The critical point of a successful attack is distinguishing between the training and testing data. Previous research suggested feasible keystone metrics, like prediction probability vector and loss [30]. The intuition is that the model is more confident in training data points than testing data points because the model is overfitting. Overfitting is the most acceptable reason for MIA but not the only factor [25, 30].

Membership Inference Attacks against GNNs. Wu et al. [1] firstly proposed MIAs to GNNs for classification tasks. Their attack methods utilized the prediction probability vector and metrics computed based on it. In contrast, He et al. [10] concentrated on MIAs against node-level GNNs. They detected the node's existence with its top two probability scores by feeding it into the model with its 0-hop or 2-hop neighborhood nodes. Olatunji et al. [17] put the overall training or testing subgraph into the model to simultaneously obtain nodes' prediction probability vectors. Those three methods are related to the prediction probability vector, which is infeasible under the label-only condition with just a prediction label as the output. This paper presents a solution for attacking GNNs without the prediction probability vector.

Label-only Membership Inference Attacks. Li et al. [14] put forward two strategies to attack traditional neural networks under the label-only condition. The first method trains a relabelled shadow model for obtaining the prediction probability vector of data points in the target dataset. The second method leverage the distance between the data point and its adversarial example to distinguish membership. In the meantime, Choquette-Choo et al. [4] put forward two approaches for MIAs under a label-only environment. One method uses the prediction correctness situation of a data point and its augmented versions for membership prediction. The other method is similar to the second approach proposed by Li et al. [14]. Those two works inspire our research of label-only MIAs against GNNs. In addition, Rahimian et al. [18] constructed the posterior vector of the data point by feeding multiple perturbed versions to the target model and analyzing the predictions of those perturbed data of this data point.

Table 7: The average attack accuracy after ten repetitions under different defenses (Cora_ML, GCN).

Row	Normalization	Dropout	Regularization	Jumping Knowledge	Avg Acc
0	×	×	×	×	0.596
1	√	×	×	×	0.586
2	×	√	×	×	0.602
3	×	×	√	×	0.573
4	×	×	×	√	0.606
5	√	√	×	×	0.609
6	√	×	√	×	0.581
7	√	×	×	√	0.615
8	×	√	√	×	0.55
9	×	√	×	√	0.607
10	×	×	√	√	0.52
11	√	√	√	×	0.609
12	√	√	×	√	0.614
13	√	×	√	√	0.618
14	×	√	√	√	0.497
15	√	√	√	√	0.613

7 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a method of implementing MIA against GNNs under the label-only condition. The average attack accuracy, precision, and AUC values of our label-only MIA are around 0.6 in most experiments, which are competitive or even better than previous probability-based MIAs implemented in the same environment and settings. In addition, we explore the influence factors of our label-only MIA. The higher overfitting level impacts the attack performance, i.e., it could increase or decrease our label-only MIA's attack accuracy. The sampling method also influences the attack performance and achieves a maximum gap of 7% on average attack accuracy. Model selection strategies with testing accuracy and loss have a slightly better attack accuracy than training accuracy and loss with a maximum gap of 1% on average attack accuracy. Then, we consider scenarios where two assumptions about the shadow dataset and the target model's information are relaxed. Surprisingly, the relaxation of those two assumptions will increase the attack performance in most experiments. Finally, we explore label-only MIA against four defense methods and their combinations. The results show that those four defenses cannot prevent our label-only MIA completely. This paper only focuses on node classification tasks. We leave label-only MIA against graph-level GNNs as feature work.

ACKNOWLEDGMENTS

This research is supported by the Chinese Scholarship Council (CSC). We are grateful to Shaofeng Li for his comments on the early proposal of this project.

AVAILABILITY

REFERENCES

[1] Wu Bang, Yang Xiangwen, Pan Shirui, and Yuan Xingliang. 2021. Adapting Membership Inference Attacks to GNN for Graph Classification: Approaches and Implications. In *Proceedings of the 21st IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7-10, 2021*. IEEE Computer Society, Los Alamitos, CA, USA, 1421–1426.

[2] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018*. OpenReview.net, Amherst, MA, USA.

[3] N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer. 2022. Membership Inference Attacks from First Principles. In *Proceedings of the 43rd IEEE Symposium on Security and Privacy, IEEE S&P 2022, San Francisco, CA, USA, May 23-25, 2022*. IEEE Computer Society, Los Alamitos, CA, USA, 1519–1519.

[4] Christopher A. Choquette-Choo, Florian Tramer, Nicholas Carlini, and Nicolas Papernot. 2021. Label-Only Membership Inference Attacks. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, Virtual Event, July 18-24, 2021*, Vol. 139. PMLR, 1964–1974.

[5] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. 2022. Data Augmentation for Deep Graph Learning: A Survey. *CoRR* abs/2202.08235 (2022).

[6] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. 2020. Quantifying Privacy Leakage in Graph Embedding. In *Proceedings of the 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous 2020, Darmstadt, Germany, December 7-9, 2020*. Association for Computing Machinery, New York, NY, USA, 76–85.

[7] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *Proceedings of the 28th World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. Association for Computing Machinery, New York, NY, USA, 417–426.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the Advances in Neural Information Processing Systems 30, NeurIPS 2017, Long Beach, CA, USA, December 4-9, 2017*, Vol. 30. Curran Associates, Inc., Red Hook, NY, USA.

[9] J. Hayes, Luca Melis, G. Danezis, and Emiliano De Cristofaro. 2019. LOGAN: Membership Inference Attacks Against Generative Models. *Proceedings on Privacy Enhancing Technologies* 2019, 1 (2019), 133–152.

[10] Xinlei He, Rui Wen, Yixin Wu, Michael Backes, Yun Shen, and Yang Zhang. 2021. Node-Level Membership Inference Attacks Against Graph Neural Networks. *CoRR* abs/2102.05429 (2021).

[11] Chenyang Hong, Qin Cao, Zhenghao Zhang, Stephen Kwok-Wing Tsui, and Kevin Y. Yip. 2021. Reusability report: Capturing properties of biological objects and their relationships using graph neural networks. *Nature Machine Intelligence* 4 (2021), 222–226. Issue 3.

[12] Bo Hui, Yuchen Yang, Haolin Yuan, Philippe Burlina, Neil Zhenqiang Gong, and Yinzhi Cao. 2021. Practical Blind Membership Inference Attack via Differential Comparisons. In *Proceedings of the 28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021*. The Internet Society, Reston, Virginia, USA.

[13] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017*. OpenReview.net, Amherst, MA, USA.

[14] Zheng Li and Yang Zhang. 2021. Membership Leakage in Label-Only Exposures. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, CCS 2021, Virtual Event, Republic of Korea, November 15 - 19, 2021*. Association for Computing Machinery, New York, NY, USA, 880–895.

[15] Yunhui Long, Lei Wang, Diyue Bu, Vincent Bindschaedler, Xiaofeng Wang, Haixu Tang, Carl A. Gunter, and Kai Chen. 2020. A Pragmatic Approach to Membership Inferences on Machine Learning Models. In *Proceedings of the 5th IEEE European Symposium on Security and Privacy, EuroS&P 2020, all-digital event, September 7-11, 2020*. 521–534.

[16] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the Advances in Neural Information Processing Systems 30, NeurIPS 2017, Long Beach, CA, USA, December 4-9, 2017*, Vol. 30. Curran Associates, Inc., Red Hook, NY, USA.

[17] Iyiola E. Olatunji, Wolfgang Nejdl, and Megha Khosla. 2021. Membership inference attack on graph neural networks. In *Proceedings of the 3rd IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, TPS-ISA 2021, Atlanta, GA, USA, December 13-15, 2021*. IEEE Computer Society, Los Alamitos, CA, USA, 11–20.

[18] Shadi Rahimian, Tribhuvanesh Orekondy, and Mario Fritz. 2021. Differential Privacy Defenses and Sampling Attacks for Membership Inference. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security, AISec 2021, Virtual Event, Republic of Korea, November 15, 2021*. Association for Computing Machinery, New York, NY, USA, 193–202.

[19] Gonzalo Martínez Ruiz de Arcaute, José Alberto Hernández, and Pedro Reviriego. 2022. Assessing the Impact of Membership Inference Attacks on Classical Machine Learning Algorithms. In *Proceedings of the 18th International Conference on the Design of Reliable Communication Networks, DRCN 2022, Vilanova i la Gelrú, Spain, March 28-31, 2022*. IEEE Computer Society, Los Alamitos, CA, USA, 1–4.

[20] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2019. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, CA, USA, February 24-27, 2019*. The Internet Society, Reston, Virginia, USA.

[21] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, He Xiaodong, and Bowen Zhou. 2019. End-to-End Structure-Aware Convolutional Networks for Knowledge Base Completion. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, Vol. 33. AAAI Press, Palo Alto, CA, USA, 3060–3067. Issue 1.

[22] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *Proceedings of the 38th IEEE Symposium on Security and Privacy, IEEE S&P 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, Los Alamitos, CA, USA, 3–18.

[23] Congzheng Song and Vitaly Shmatikov. 2019. Auditing Data Provenance in Text-Generation Models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. Association for Computing Machinery, New York, NY, USA, 196–206.

[24] Liwei Song and Prateek Mittal. 2021. Systematic Evaluation of Privacy Risks of Machine Learning Models. In *Proceedings of the 30th USENIX Security Symposium, USENIX Security 2021, Vancouver, B.C., Canada, August 11-13, 2021*, Michael Bailey and Rachel Greenstadt (Eds.). USENIX Association, 2615–2632.

[25] Stacey Truex, Ling Liu, Mehmet Emre Gursoy, Lei Yu, and Wenqi Wei. 2018. Towards Demystifying Membership Inference Attacks. *CoRR* abs/1807.09173 (2018).

[26] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018*. OpenReview.net, Amherst, MA, USA.

[27] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. Association for Computing Machinery, New York, NY, USA, 968–977.

[28] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, Amherst, MA, USA.

[29] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, Vol. 80. PMLR, 5449–5458.

[30] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *Proceedings of the 31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*. IEEE Computer Society, Los Alamitos, CA, USA, 268–282.

[31] Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. 2019. Attributed Graph Clustering via Adaptive Graph Convolution. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. International Joint Conferences on Artificial Intelligence Organization, CA, USA, 4327–4333.

[32] Zhikun Zhang, Min Chen, Michael Backes, Yun Shen, and Yang Zhang. 2022. Inference Attacks Against Graph Neural Networks. In *Proceedings of the 31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10–12, 2022*. USENIX Association, Berkeley, CA, USA.

A ADDITIONAL RESULTS

Table 8 indicates the attack performance of four previous probability-based MIAs with classifiers, including 0-hop, 2-hop, the combination of 0-hop and 2-hop, and all probability methods. We implement those four MIAs under the same settings as the results of Table 2. The settings include the same target and shadow models (a low overfitting level) trained with the same dataset split, the training of the attack models, and the model selection strategy. Each result

Table 8: The attack performance of four probability-based attacks with classifiers after ten repetitions.

Dataset	GNN	probability-based methods with classifiers (avg acc, pre, rec, auc, f1, low_fpr_0.01_tpr)			
		0-hop	2-hop	0-hop and 2-hop combination	all probability
Cora_ML	GAT	[**0.576**, 0.552, 0.634, 0.626, 0.546, 0.076]	[**0.549**, 0.508, 0.457, 0.61, 0.409, 0.033]	[**0.588**, 0.563, 0.62, 0.629, 0.548, 0.088]	[**0.571**, 0.511, 0.493, 0.652, 0.454, 0.078]
	GCN	[**0.691**, 0.67, 0.874, 0.765, 0.746, 0.157]	[**0.633**, 0.622, 0.807, 0.685, 0.687, 0.059]	[**0.693**, 0.675, 0.851, 0.765, 0.739, 0.142]	[**0.635**, 0.681, 0.639, 0.723, 0.591, 0.121]
	GIN	[**0.613**, 0.611, 0.761, 0.692, 0.655, 0.092]	[**0.575**, 0.577, 0.722, 0.604, 0.609, 0.009]	[**0.6**, 0.58, 0.724, 0.666, 0.629, 0.106]	[**0.55**, 0.587, 0.52, 0.575, 0.51, 0.026]
	GraphSAGE	[**0.627**, 0.68, 0.732, 0.747, 0.636, 0.136]	[**0.626**, 0.685, 0.73, 0.762, 0.625, 0.107]	[**0.639**, 0.66, 0.793, 0.74, 0.68, 0.124]	[**0.61**, 0.66, 0.324, 0.637, 0.372, 0.115]
CiteSeer	GAT	[**0.533**, 0.478, 0.329, 0.562, 0.338, 0.013]	[**0.518**, 0.422, 0.436, 0.539, 0.399, 0.009]	[**0.524**, 0.426, 0.467, 0.555, 0.414, 0.014]	[**0.515**, 0.38, 0.281, 0.531, 0.295, 0.01]
	GCN	[**0.555**, 0.456, 0.45, 0.567, 0.439, 0.011]	[**0.559**, 0.523, 0.475, 0.6, 0.463, 0.017]	[**0.551**, 0.526, 0.425, 0.575, 0.443, 0.009]	[**0.528**, 0.507, 0.366, 0.525, 0.388, 0.007]
	GIN	[**0.535**, 0.48, 0.618, 0.562, 0.508, 0.01]	[**0.541**, 0.535, 0.759, 0.567, 0.601, 0.011]	[**0.534**, 0.534, 0.728, 0.568, 0.581, 0.01]	[**0.515**, 0.498, 0.248, 0.519, 0.266, 0.01]
	GraphSAGE	[**0.522**, 0.476, 0.411, 0.534, 0.348, 0.014]	[**0.54**, 0.416, 0.381, 0.551, 0.354, 0.012]	[**0.53**, 0.417, 0.393, 0.569, 0.364, 0.011]	[**0.519**, 0.312, 0.26, 0.523, 0.268, 0.01]
DBLP	GAT	[**0.516**, 0.539, 0.175, 0.547, 0.231, 0.011]	[**0.514**, 0.471, 0.276, 0.539, 0.278, 0.011]	[**0.517**, 0.479, 0.274, 0.54, 0.299, 0.01]	[**0.503**, 0.52, 0.161, 0.521, 0.188, 0.015]
	GCN	[**0.532**, 0.545, 0.406, 0.561, 0.415, 0.011]	[**0.534**, 0.548, 0.424, 0.556, 0.419, 0.013]	[**0.532**, 0.543, 0.412, 0.562, 0.421, 0.011]	[**0.527**, 0.524, 0.416, 0.549, 0.432, 0.014]
	GIN	[**0.514**, 0.436, 0.325, 0.531, 0.299, 0.012]	[**0.517**, 0.499, 0.373, 0.529, 0.339, 0.006]	[**0.524**, 0.453, 0.358, 0.538, 0.318, 0.006]	[**0.507**, 0.372, 0.277, 0.527, 0.266, 0.006]
	GraphSAGE	[**0.531**, 0.53, 0.49, 0.553, 0.465, 0.011]	[**0.538**, 0.537, 0.474, 0.558, 0.463, 0.012]	[**0.539**, 0.548, 0.391, 0.562, 0.423, 0.015]	[**0.554**, 0.557, 0.576, 0.583, 0.536, 0.017]
PubMed	GAT	[**0.498**, 0.415, 0.185, 0.503, 0.198, 0.008]	[**0.5**, 0.341, 0.384, 0.497, 0.306, 0.009]	[**0.497**, 0.442, 0.365, 0.501, 0.352, 0.009]	[**0.505**, 0.464, 0.531, 0.511, 0.456, 0.01]
	GCN	[**0.506**, 0.353, 0.217, 0.515, 0.229, 0.009]	[**0.506**, 0.478, 0.291, 0.512, 0.286, 0.009]	[**0.504**, 0.446, 0.203, 0.508, 0.224, 0.008]	[**0.507**, 0.476, 0.342, 0.516, 0.353, 0.007]
	GIN	[**0.502**, 0.393, 0.649, 0.508, 0.455, 0.007]	[**0.502**, 0.398, 0.376, 0.508, 0.294, 0.007]	[**0.501**, 0.267, 0.375, 0.504, 0.262, 0.009]	[**0.499**, 0.239, 0.125, 0.497, 0.134, 0.006]
	GraphSAGE	[**0.504**, 0.404, 0.273, 0.513, 0.307, 0.008]	[**0.508**, 0.403, 0.295, 0.521, 0.302, 0.009]	[**0.504**, 0.451, 0.293, 0.52, 0.338, 0.009]	[**0.511**, 0.416, 0.334, 0.525, 0.351, 0.008]

Table 9: The attack performance of transfer attack and four probability-based attacks with metrics after ten repetitions.

Dataset	GNN	Test Acc (target model)	Train Acc (target model)	Attack performance (avg acc, pre, rec, auc, f1, low_fpr_0.01_tpr) '_' means that this setting does not have a corresponding metric				
				label-only method	probability-based methods with metrics			
				Transfer Attack	Gap Attack	Probability of Ground Truth	Cross-Entropy	Modified Cross-Entropy
Cora_ML	GAT	0.722	0.855	[**0.516**, 0.532, 0.288, 0.526, 0.367, 0.02]	**0.567**, 0.542, 0.855, _, 0.664, _	[**0.506**, 0.503, 0.994, 0.731, 0.668, 0.191]	[**0.502**, 0.4, 0.197, 0.274, 0.138, 0.014]	[**0.486**, 0.453, 0.813, 0.269, 0.556, 0.007]
	GCN	0.753	0.95	[**0.628**, 0.63, 0.625, 0.685, 0.625, 0.081]	[**0.598**, 0.558, 0.95, _, 0.703, _]	[**0.51**, 0.505, 0.997, 0.818, 0.671, 0.232]	[**0.57**, 0.623, 0.231, 0.196, 0.24, 0.018]	[**0.424**, 0.422, 0.774, 0.182, 0.54, 0.001]
	GIN	0.673	0.877	[**0.515**, 0.54, 0.234, 0.521, 0.32, 0.017]	[**0.602**, 0.566, 0.877, _, 0.687, _]	[**0.51**, 0.505, 0.993, 0.726, 0.669, 0.024]	[**0.517**, 0.259, 0.146, 0.301, 0.132, 0.009]	[**0.472**, 0.469, 0.861, 0.274, 0.598, 0.003]
	GraphSAGE	0.719	0.957	[**0.524**, 0.551, 0.259, 0.535, 0.35, 0.02]	[**0.619**, 0.571, 0.957, _, 0.715, _]	[**0.514**, 0.507, 0.999, 0.819, 0.673, 0.215]	[**0.557**, 0.586, 0.265, 0.21, 0.239, 0.021]	[**0.429**, 0.409, 0.737, 0.181, 0.516, 0.0]
CiteSeer	GAT	0.798	0.895	[**0.501**, 0.501, 0.492, 0.505, 0.495, 0.009]	[**0.549**, 0.529, 0.895, _, 0.665, _]	[**0.506**, 0.503, 0.996, 0.583, 0.668, 0.006]	[**0.513**, 0.157, 0.139, 0.434, 0.129, 0.005]	[**0.485**, 0.481, 0.865, 0.417, 0.606, 0.004]
	GCN	0.8	0.936	[**0.512**, 0.511, 0.521, 0.514, 0.515, 0.011]	[**0.568**, 0.539, 0.936, _, 0.684, _]	[**0.501**, 0.501, 1.0, 0.603, 0.667, 0.008]	[**0.498**, 0.113, 0.013, 0.422, 0.022, 0.006]	[**0.502**, 0.501, 0.987, 0.397, 0.665, 0.001]
	GIN	0.779	0.915	[**0.513**, 0.514, 0.521, 0.517, 0.516, 0.011]	[**0.568**, 0.54, 0.915, _, 0.679, _]	[**0.511**, 0.506, 0.998, 0.591, 0.671, 0.004]	[**0.502**, 0.101, 0.121, 0.437, 0.098, 0.006]	[**0.489**, 0.464, 0.882, 0.409, 0.601, 0.002]
	GraphSAGE	0.794	0.943	[**0.509**, 0.509, 0.53, 0.512, 0.517, 0.012]	[**0.574**, 0.543, 0.943, _, 0.689, _]	[**0.51**, 0.505, 0.999, 0.617, 0.671, 0.009]	[**0.523**, 0.227, 0.242, 0.406, 0.224, 0.005]	[**0.468**, 0.452, 0.762, 0.383, 0.551, 0.001]
DBLP	GAT	0.737	0.828	[**0.508**, 0.508, 0.523, 0.509, 0.515, 0.011]	[**0.546**, 0.529, 0.828, _, 0.646, _]	[**0.502**, 0.501, 0.996, 0.577, 0.667, 0.01]	[**0.505**, 0.121, 0.06, 0.44, 0.058, 0.007]	[**0.493**, 0.494, 0.945, 0.423, 0.644, 0.006]
	GCN	0.752	0.886	[**0.513**, 0.512, 0.552, 0.518, 0.531, 0.011]	[**0.567**, 0.541, 0.886, _, 0.672, _]	[**0.501**, 0.5, 1.0, 0.606, 0.667, 0.008]	[**0.499**, 0.111, 0.009, 0.423, 0.016, 0.007]	[**0.501**, 0.5, 0.991, 0.394, 0.665, 0.003]
	GIN	0.721	0.863	[**0.511**, 0.511, 0.507, 0.514, 0.509, 0.012]	[**0.571**, 0.545, 0.863, _, 0.668, _]	[**0.501**, 0.5, 1.0, 0.587, 0.667, 0.005]	[**0.498**, 0.048, 0.003, 0.457, 0.005, 0.004]	[**0.502**, 0.501, 0.997, 0.413, 0.667, 0.003]
	GraphSAGE	0.733	0.892	[**0.511**, 0.511, 0.543, 0.514, 0.526, 0.012]	[**0.579**, 0.549, 0.892, _, 0.68, _]	[**0.506**, 0.503, 0.997, 0.624, 0.669, 0.01]	[**0.502**, 0.24, 0.108, 0.411, 0.085, 0.005]	[**0.49**, 0.471, 0.898, 0.376, 0.609, 0.002]
PubMed	GAT	0.859	0.885	[**0.506**, 0.506, 0.539, 0.506, 0.517, 0.01]	[**0.513**, 0.508, 0.885, _, 0.645, _]	[**0.502**, 0.501, 0.987, 0.53, 0.665, 0.008]	[**0.503**, 0.251, 0.234, 0.477, 0.2, 0.009]	[**0.494**, 0.489, 0.782, 0.47, 0.571, 0.008]
	GCN	0.868	0.905	[**0.505**, 0.504, 0.61, 0.505, 0.551, 0.01]	[**0.519**, 0.51, 0.905, _, 0.653, _]	[**0.502**, 0.501, 0.994, 0.529, 0.666, 0.009]	[**0.503**, 0.252, 0.172, 0.482, 0.148, 0.008]	[**0.495**, 0.491, 0.837, 0.471, 0.596, 0.005]
	GIN	0.849	0.895	[**0.508**, 0.507, 0.575, 0.508, 0.537, 0.01]	[**0.523**, 0.513, 0.895, _, 0.652, _]	[**0.504**, 0.502, 0.989, 0.527, 0.666, 0.006]	[**0.502**, 0.102, 0.175, 0.488, 0.128, 0.007]	[**0.494**, 0.484, 0.837, 0.473, 0.583, 0.005]
	GraphSAGE	0.867	0.92	[**0.507**, 0.506, 0.599, 0.508, 0.548, 0.01]	[**0.526**, 0.515, 0.92, _, 0.66, _]	[**0.504**, 0.502, 0.994, 0.547, 0.667, 0.007]	[**0.507**, 0.234, 0.213, 0.467, 0.194, 0.008]	[**0.489**, 0.483, 0.794, 0.453, 0.579, 0.004]

row in Table 8 corresponds with the result row under the same dataset and GNN model in Table 2. For each row, the dataset and GNN are used to train the target and shadow models. The "0-hop", "2-hop", "o-hop and 2-hop combination", and "all probability" in the table indicate the results of four probability-based MIAs with classifiers. Each result has six values: the average accuracy, precision, recall, AUC value, F1 score, and TPR under low FPR (0.01) after ten repetitions. We highlight the average accuracy of each result for a clear comparison in bold and red.

Table 9 presents the attack performance of the transfer attack (Li et al. [14]) and four previous probability-based MIAs with metrics. Four probability-based MIAs with metrics utilize prediction correctness (Gap Attack), probability of ground truth, cross-entropy, and modified cross-entropy for determining membership by comparing with a threshold [20, 22, 24, 30]. The evaluation metrics are the same as Table 8. The "_" in the table means we cannot obtain this metric because no threshold is related to metric calculation.