# NODE-LEVEL MEMBERSHIP INFERENCE ATTACKS AGAINST GRAPH NEURAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Many real-world data are graphs, such as social networks and protein structures. To fully utilize the information contained in graph data, graph neural networks (GNNs) have been introduced. Previous studies have shown that machine learning models are vulnerable to privacy attacks. However, most of the current efforts concentrate on ML models trained on images and texts. On the other hand, privacy risks stemming from GNNs remain largely unstudied. In this paper, we fill the gap by performing the first comprehensive analysis of node-level membership inference attacks against GNNs. We systematically define the threat models and propose eight node-level membership inference attacks based on an adversary's background knowledge. Our evaluation on four GNN structures and four benchmark datasets shows that GNNs are vulnerable to node-level membership inference even when the adversary has minimal background knowledge. Besides, we show that node degree, graph density, and feature similarity have major impacts on the attack's success. We further investigate three defense mechanisms and show that differential privacy (DP) can better protect the membership privacy while preserving the model's utility.

## 1 INTRODUCTION

Many real-world data can be organized in the form of graphs, such as social relations and protein structures. Effective graph analysis provides users a deeper understanding of what is behind the data and can help to analyze many natural phenomena and build powerful commercial applications. To fully utilize the rich information of graph data, a new family of machine learning (ML) models, namely graph neural networks (GNNs) (Kipf & Welling, 2017; Hamilton et al., 2017; Velickovic et al., 2018), has been introduced to address graph-related tasks in an end-to-end manner. GNN models utilize both the feature of each sample and the features of the sample's neighborhood to represent the sample. In this way, a GNN learns to embed the structural connections among different nodes in its training dataset.

Recent research has shown that ML models are vulnerable to privacy attacks (Shokri et al., 2017; Salem et al., 2019; Fredrikson et al., 2015; Leino & Fredrikson, 2020; Carlini et al., 2019; Melis et al., 2019; Song & Shmatikov, 2019; 2020). Most of the current efforts in this direction concentrate on ML models trained on sensitive non-structured data, such as images and texts. Meanwhile, graph data, which is used to train GNNs, also contains sensitive information, such as social relations (Backstrom et al., 2007; Crandall et al., 2010; Jia et al., 2017) and mobility traces (Cho et al., 2011; Backes et al., 2017). However, the potential privacy risks stemming from GNNs have been largely understudied. There exists some preliminary work on node-level membership inference attacks against GNNs (Duddu et al., 2020; Olatunji et al., 2021). However, the early demonstrations of the attack's vulnerability have a strong assumption whereby the adversary has the full neighborhood information, which is a rather strong assumption that limits the scope of meaningful membership inference attacks against GNNs.

In this paper, we investigate whether a GNN model is vulnerable to membership inference attacks (Shokri et al., 2017; Salem et al., 2019; Song & Mittal, 2021), which are the major means to assess ML models' privacy risks. Specifically, an adversary aims to infer whether a target node is used to train a target GNN. We concentrate on black-box membership inference, the most difficult setting for the adversary (Shokri et al., 2017). As mentioned before, GNNs are designed for graph

data that is not in the Euclidean space, which leads to some unique research questions for membership inference attacks in this setting. First, an adversary needs background knowledge, such as the target GNN's architecture and a shadow dataset, to train their attack model. Also, different graphs share many common properties, such as power-law degree distribution (Leskovec et al., 2014). This motivates us to understand whether an adversary can have less constrained background knowledge compared to previous membership inference attacks against other types of ML models. Second, an adversary can query a node to a GNN with either the node's feature alone or the node and its neighborhood's graph connections as well as their features. This means that one node can receive different prediction outputs (posteriors) from the GNN. We are interested in which posteriors reveal more information of the target node's membership status and whether these posteriors can be combined to achieve a more effective attack. To answer these research questions, we make the following contributions in this paper.

We first systematically define the threat model of node-level membership inference attack against GNNs by categorizing an adversary's background knowledge along three dimensions, i.e., shadow dataset, shadow model, and node topology. Following the different threat models based on node topology, we propose four membership inference attack models, namely 0-hop, 1-hop, 2-hop, and combined attacks. Different from previous work (Carlini et al., 2021; Watson et al., 2021), we develop a new difficulty calibration method based on Jensen-Shannon distance (JS distance) which can further facilitate our attacks and do not need extra reference models.

We perform an extensive evaluation on four popular GNN models with four benchmark datasets. Experimental results show that our attacks achieve strong performance. More interestingly, we discover that our 0-hop attack has better performance than the 1-hop and 2-hop attacks, which is overlooked by previous work (Duddu et al., 2020; Olatunji et al., 2021). This is because a target node's 1-hop or 2-hop neighborhood contains a mixture of member and non-member nodes which jeopardizes the attack model's accuracy. Our combined attack is more effective by taking advantage of the 0-hop, 1-hop, and 2-hop attacks. Also, the calibrated version of previous attacks can further breach the node-level membership privacy. Moreover, we show that our attacks are still effective when the adversary does not know the target model's training dataset distribution or architecture.

We also perform an in-depth analysis of what kind of node is more vulnerable to the attack. Our experiments reveal that a node with higher degree is more robust to membership inference as the GNN aggregates more neighbor's information this reduce the "exposure" of the node itself. Also, a node with a higher subgraph density is more prone to membership inference, since a dense subgraph drives the node to participate more in the aggregation process of the GNN training, which amplifies the node's influence in the target GNN model. Besides, it is easier for the adversary to mount their attack if a node shares similar features with its neighbors (see Section 4.2). To mitigate the attacks, we evaluate three defense mechanisms, i.e., random edge addition, label-only output, and differential privacy (DP). Empirical evaluation shows that DP can achieve a better trade-off between the model utility and the membership privacy.

## 2 PRELIMINARY

**Notations.** We define a graph dataset as $\mathcal{D} = (\mathcal{G}, \mathcal{X}, \mathcal{Y})$. Here, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represents a graph with $\mathcal{V}$ denoting the graph's set of nodes and $\mathcal{E}$ representing the set of edges connecting these nodes. Each node is denoted by $v \in \mathcal{V}$ and $e_{uv} \in \mathcal{E}$ represents an edge linking two nodes $u$ and $v$. $\mathcal{X} = \{x_1, x_2, ..., x_{|\mathcal{V}|}\}$ and $\mathcal{Y} = \{y_1, y_2, ..., y_{|\mathcal{V}|}\}$ represent the features and labels for all the nodes in $\mathcal{G}$, respectively. Node $v$'s $l$-hop neighborhood is denoted by $\mathcal{N}^l(v)$, which contains a set of nodes at a distance less than or equal to $l$ from $v$ in $\mathcal{G}$. For convenience, we abbreviate the 1-hop neighborhood of $v$ as $\mathcal{N}(v)$. The $l$-hop subgraph of node $v$, denoted by $g^l(v)$, contains $v$, its $l$-hop neighborhood $\mathcal{N}^l(v)$, edges among these nodes, and features of these nodes.

**GNN Architecture.** Basically, a GNN contains multiple graph convolution layers that iteratively update a node $v$'s representation by aggregating the representation of nodes in $v$'s neighborhood. Formally, each graph convolution layer of a GNN model can be defined as: $z_v^{(t)} = \mathsf{AGGREGATE}(h_v^{(t-1)}, \{h_u^{(t-1)} : u \in \mathcal{N}(v)\})$ and $h_v^{(t)} = \mathsf{UPDATE}(z_v^{(t)})$. Here $\mathcal{N}(v)$ is the neighborhood of $v$. $t$ represents the $t$-th layer of the GNN. $z_v^{(t)}$ and $h_v^{(t)}$ denote the hidden state and the representation vector of node $v$ at layer $t$. In the first step, we initialize $v$'s representation $h_v^{(0)}$ as

its feature $x_v$. $\mathsf{AGGREGATE}(\cdot)$ and $\mathsf{UPDATE}(\cdot)$ are the aggregation and update functions, respectively. Given a node $v$, the aggregation function is used to generate the current hidden state $z_v^{(t)}$ using a combination of its previous representation and the aggregated representation from its neighborhood $\mathcal{N}(v)$. The update function then conducts non-linear transformation on the current hidden state $z_v^{(t)}$ and produces the representation vector $h_v^{(t)}$. In this paper, we focus on four representative GNN architectures, i.e., GraphSAGE (Hamilton et al., 2017), GAT (Velickovic et al., 2018), GIN (Xu et al., 2019), and GCN (Kipf & Welling, 2017).

**GNN Prediction.** In this paper, we focus on node classification tasks. In the training phase, an inductive GNN learns the parameters of aggregation and update functions in different layers over a training dataset. Then, to get a precise prediction of an unseen node $v$ in a $t$-layer GNN, we can feed $v$'s $t$-hop subgraph, i.e., $g^t(v)$, to the GNN and obtain the prediction posteriors $p_v$. Note that the $t$-hop subgraph of $v$ is not a necessary condition to acquire the posteriors $p_v$. We can obtain posteriors $p_v$ by only querying the target node $v$'s feature to the GNN model. Our evaluation shows that even in this case, the GNN model can achieve better performance than MLP, i.e., a model that does not consider graph structural information (see Figure 8 in Appendix).

## 3 Node-level Membership Inference Attacks

### 3.1 Problem Definition

The adversary aims to determine whether a given node is used to train a target GNN model or not. More formally, given a target node $v$, a target GNN model $\mathcal{M}_T$, and the adversary's background knowledge $\mathcal{K}$, node-level membership inference attack $\mathcal{A}$ can be defined as: $\mathcal{A} : v, \mathcal{M}_T, \mathcal{K} \mapsto \{\textit{member}, \textit{non-member}\}$. Successful membership inference attacks can cause severe privacy risks (Shokri et al., 2017; Salem et al., 2019). In the setting of GNNs, membership threat is related to graph data, such as inferring a user being a member of a sensitive social network or a patient network (e.g., AIDS/COVID contact trace network), a transaction being part of a transaction graph, etc. Consequently, successful membership inference can reveal sensitive information (sexual orientation, spending behavior, etc) when the GNN model is trained on data from a specific subgroup of the population.

### 3.2 Threat Model

Our target model $\mathcal{M}_T$ is an inductive GNN model. First, we assume that the adversary only has black-box access to the target model, i,e, they can only query the target model and obtain the posteriors. As mentioned by previous work (Shokri et al., 2017; Salem et al., 2019; He et al., 2021), black-box setting is the most challenging scenario for the adversary. We then categorize the adversary's background knowledge $\mathcal{K}$ along three dimensions, i.e., shadow dataset, shadow model, and node topology.

**Shadow Dataset.** We assume that the adversary has a shadow dataset $\mathcal{D}_{\textit{Shadow}}$ which contains its own graph structure as well as node features and labels. Following the previous work (Shokri et al., 2017), the shadow dataset $\mathcal{D}_{\textit{Shadow}}$ can come from the same distribution of the target model's training dataset. However, our empirical evaluation shows that this assumption can be relaxed (see Section 4.4). Note that in both cases, the shadow dataset has no node and edge intersection with the target dataset.

**Shadow Model.** With the shadow dataset, the adversary can train a shadow GNN model $\mathcal{M}_S$ to mimic the target model $\mathcal{M}_T$. We can assume that the shadow model shares the same architecture as the target model (Shokri et al., 2017; Salem et al., 2019). However, our experimental results show that an adversary can use a different GNN architecture from the target model to establish their shadow model (see Section 4.4).

**Node Topology.** To get the posteriors for $v$ from $\mathcal{M}_T$, we consider three cases. In the first case, we assume that the adversary only has $v$'s feature $x_v$. As the input to a GNN needs to be in the form of a graph, we add a self-loop for $v$ (Kipf & Welling, 2017) and query the target model. We refer to this case as a node's *0-hop query*. In the second case, we assume that the adversary knows the target node $v$'s 1-hop subgraph $g^1(v)$, which can be directly fed to the target model. We refer to this case

as a node's *1-hop query*. In the third case, similar to the second one, we assume that the adversary knows the target node $v$'s 2-hop subgraph $g^2(v)$. We name this scenario as a node's *2-hop query*. Note that the subgraph does not need to be the complete subgraph of $v$ as the adversary may only have a partial view of the dataset. Besides, nodes in the subgraph can be a mixture of members and non-members for the target GNN. This is more realistic as the adversary does not know any other nodes' membership status. The goal is to infer the membership status of $v$. In general, we consider the 0-hop, 1-hop, and 2-hop queries in this paper. Note that *the adversary can choose $k$-hop query with $k > 2$*. However, most of the state-of-the-art GNNs follow a two-layer structure due to the fact that real-world graphs normally exhibit small-world phenomenon (Easley & Kleinberg, 2010), and in this case, 2-hop query is the upper bound for the query depth. In another way, $k$-hop query ($k > 2$) will be considered as a 2-hop query for a two-layer GNN. Moreover, previous empirical results (Hamilton et al., 2017) show that deeper GNN architecture does not further improve the classification performance. Therefore, we only consider $k$-hop query ($k \leq 2$) in this paper and leave $k$-hop query ($k > 2$) as the future work.

### 3.3 Attack Methodology

Following the standard process of membership inference attacks against ML models (Shokri et al., 2017), our attack can be divided into three stages, i.e., shadow model training, attack model training, and membership inference. Figure 9 (in Appendix A) provides a schematic overview of the attack process.

**Shadow Model Training.** Given a shadow dataset $\mathcal{D}_{Shadow}$, the adversary first splits its node set $\mathcal{V}_{Shadow}$ into two disjoint sets $\mathcal{V}_{Shadow}^{Train}$ and $\mathcal{V}_{Shadow}^{Test}$. Then, the adversary derives their shadow training ($\mathcal{D}_{Shadow}^{Train}$) and testing ($\mathcal{D}_{Shadow}^{Test}$) datasets by involving all the features, labels, and edges within $\mathcal{V}_{Shadow}^{Train}$ and $\mathcal{V}_{Shadow}^{Test}$, respectively. After that, $\mathcal{D}_{Shadow}^{Train}$ is used to train a shadow GNN model $\mathcal{M}_S$.

**Attack Model Training.** The attack model is a binary classifier and its input is derived from a node's posteriors provided by a GNN. To obtain the training dataset for the attack model, the adversary needs to query $\mathcal{M}_S$ with all the nodes in $\mathcal{V}_{Shadow}$ (both $\mathcal{V}_{Shadow}^{Train}$ and $\mathcal{V}_{Shadow}^{Test}$) and gets the corresponding prediction posteriors. As mentioned before, depending on their knowledge of node topology, the adversary can perform 0-hop query, 1-hop query, or 2-hop query. For a node $v$, we refer to its posteriors obtained by $k$-hop query as *$k$-hop posteriors*. Inspired by previous work (Carlini et al., 2021; Watson et al., 2021), we propose a new difficulty calibration method to facilitate membership inference attacks but without extra reference models. Concretely, given a model where $v$ belongs to its training dataset, we need a reference model where $v$ does not belong to its training dataset. Instead of training multiple reference models (Carlini et al., 2021; Watson et al., 2021), we consider the shadow model and target model as each other's reference model since their training datasets have no intersection. Then we define the JS distance between the posteriors obtained from both models as the *difficulty level* of $v$. The intuition is that $v$ is an easy-to-predict node if both model predict $v$ with very high similarity.

In this paper, we first consider four types of attack model input summarized from posteriors which leads to four attack models, namely 0-hop attack $\mathcal{A}_0$, 1-hop attack $\mathcal{A}_1$, 2-hop attack $\mathcal{A}_2$, and combined attack $\mathcal{A}_c$. For each attack model $\mathcal{A}$, we then consider an calibrated version $\mathcal{A}^{cal}$ by adding the difficulty level into the input as well. In total, we have eight attack models.

*0/1/2-hop Attacks ($\mathcal{A}_0/\mathcal{A}_1/\mathcal{A}_2$).* The 0/1/2-hop attack models are essentially MLPs, which take $v$'s largest two[1] values (ranked) in its 0/1/2-hop posteriors as the input.

*Combined Attack ($\mathcal{A}_c$).* The combined attack model considers both the inputs for the 0-hop, 1-hop, and the 2-hop attack by first feeding them separately to different linear layers. Then, the attack model concatenates the three embeddings and feeds them to an MLP.

*Calibrated Attack ($\mathcal{A}^{cal}$).* The calibrated attack model follows the same architecture as the previous attacks but add also the difficulty level into the input. E.g., for the combined attack $\mathcal{A}_c$, its calibrated version $\mathcal{A}_c^{cal}$ consider the difficulty level for 0/1/2-hop posteriors simultaneously.

Note that $\mathcal{A}_0$ and $\mathcal{A}_1$ are practically important. For instance, social networks such as Instagram or Tinder do not reveal social relationships of (private) user accounts. However, an adversary can crawl

[1]Classification tasks considered in this paper have at least two classes.

users' profile information without or with limited social relations. They can still launch membership inference attacks (i.e., $\mathcal{A}_0$ and $\mathcal{A}_1$). Besides, if the adversary can perform 2-hop attack of a given node, they can also perform 0-hop and 1-hop attack. Therefore, the combined attack requires the same background knowledge as the 2-hop attack. In all cases, if $v \in \mathcal{V}_{Shadow}^{Train}$, we label it as a member, otherwise as a non-member. In the end, the adversary constructs an attack training dataset, which they use to train their attack model.

**Membership Inference.** To determine whether a target node is used to train the target model $\mathcal{M}_T$, the adversary first conducts $k$-hop query to the target model depending on their background knowledge and obtains the corresponding posteriors. Then, the adversary queries the attack model with the posteriors to get the node's membership prediction.

# 4 EVALUATION

## 4.1 EXPERIMENTAL SETUP

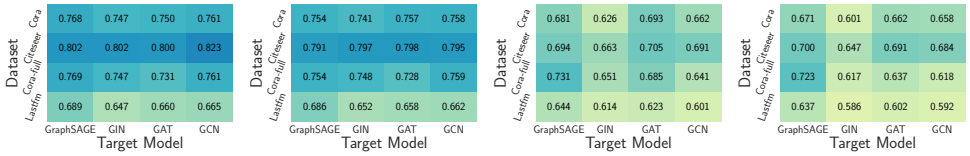
**Dataset.** We conduct experiments on four public datasets, including Cora (Kipf & Welling, 2017), Citeseer (Kipf & Welling, 2017), Cora-full (Bojchevski & Günnemann, 2018), and LastFM Asia (Rozemberczki & Sarkar, 2020) (abbreviated as Lastfm). Cora and Citeseer are citation graphs whose nodes represent papers and edges reflect citation relationships among papers. Cora-full is an extended Cora dataset. Lastfm is a social network dataset with its nodes being users and edges representing users' mutual following relationships. All datasets contain node features and labels. Dataset statistics are summarized in Table 1 in Appendix A.

**Dataset Configuration.** The dataset configuration process is depicted in Figure 9. For each dataset, we first randomly split its nodes by half. The first half (including the nodes, the edges among the nodes, and the nodes' features and labels) is used to construct the target dataset, i.e., $\mathcal{D}_{Target}$. The other half is treated as the shadow dataset, i.e., $\mathcal{D}_{Shadow}$. Note that the target dataset and shadow dataset are disjoint as mentioned in Section 3. For the target dataset $\mathcal{D}_{Target}$, we further randomly split it by half creating the target training dataset $\mathcal{D}_{Target}^{Train}$ and the target testing dataset $\mathcal{D}_{Target}^{Test}$. The target training dataset is used to train the target model, and the target testing dataset is used to test the target model's performance with respect to its original classification task. Both $\mathcal{D}_{Target}^{Train}$ and $\mathcal{D}_{Target}^{Test}$ are used to test membership inference. Nodes in $\mathcal{D}_{Target}^{Train}$ are considered as members and nodes in $\mathcal{D}_{Target}^{Test}$ as non-members. As mentioned in Section 3.2, for the 2-hop query scenario, each node's 2-hop subgraph can contain a mixture of member and non-member nodes.

We apply the same processing procedure on the shadow dataset to generate the shadow training dataset $\mathcal{D}_{Shadow}^{Train}$ and the shadow testing dataset $\mathcal{D}_{Shadow}^{Test}$. $\mathcal{D}_{Shadow}^{Train}$ is used to train the shadow model. Both $\mathcal{D}_{Shadow}^{Train}$ and $\mathcal{D}_{Shadow}^{Test}$ are used to derive the training dataset for the attack model. We use accuracy as our evaluation metric for both target model's performance and attack model's performance as it is widely used in node classification tasks (Kipf & Welling, 2017; Velickovic et al., 2018; Xu et al., 2019) as well as membership inference attacks (Shokri et al., 2017; Salem et al., 2019). We consider 2-hop attack as the baseline attack as Duddu et al. (2020) and Olatunji et al. (2021) leverage 2-hop attack as their attack methodology. The training details of target and attack models are summarized in Section A.1.

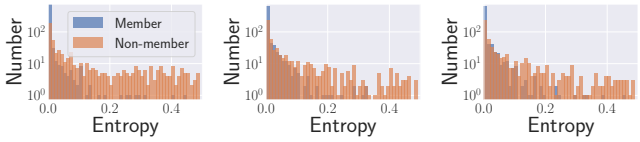
## 4.2 0-HOP, 1-HOP, 2-HOP, AND COMBINED ATTACKS

We first show the membership inference attack performance of the combined, 0-hop, 1-hop, and 2-hop attacks in Figure 1. We find that compared to the 1-hop and 2-hop attacks, the 0-hop attack achieves higher membership inference accuracy. Also, the 1-hop attack performs better than the 2-hop attack. For instance, the baseline attack (2-hop) on GraphSAGE trained on Cora only achieves 0.671 accuracy while the accuracy of the corresponding 1-hop and 0-hop attack are 0.681 and 0.754, respectively. Such observations reveal that a node's 0-hop or 1-hop query to the target GNN leaks more membership information of the node although such a query leverage less information compared to the 2-hop query, which is overlooked by previous work (Duddu et al., 2020; Olatunji et al., 2021).




(a) Combined (b) 0-hop (c) 1-hop (d) 2-hop (Baseline)

Figure 1: The performance of combined, 0-hop, 1-hop, and 2-hop attacks for different GNN architectures on four different datasets.



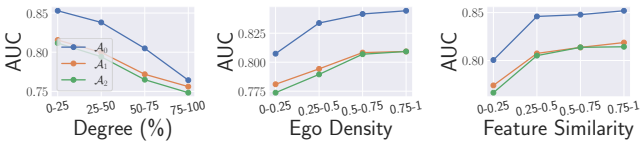

(a) 0-hop (b) 1-hop (c) 2-hop

Figure 2: The entropy distribution of posteriors from 0-hop, 1-hop, and 2-hop queries for GraphSAGE on Citeseer.

To investigate the reason behind this, we visualize the entropy distribution of posteriors from 0/1/2-hop queries for GrapgSage on Citeseer and the results are summarized in Figure 2. We observe that, compared to 1-hop or 2-hop queries, the entropy from 0-hop query is distributed more separately. For instance, the JS-Divergence between member and non-member's entropy distribution of posteriors are 0.325, 0.233, and 0.232 from 0-hop, 1-hop, and 2-hop queries. A larger value indicates a lower difficulty to differentiate them. It is reasonable since if a target node is a member, its 1-hop or 2-hop subgraph may contain some non-member nodes, which yields a less confident prediction and makes it harder to be separated from non-members. Note that You et al. (2021) also show that the target node's information is less pronounced in a GNN's aggregated outputs.

This can also be credited to different overfitting level of the target model (the gap between training and testing accuracy). Given the overfitting level $g$, a lower bound of membership inference attack accuracy is $\frac{1+g}{2}$ (see Theorem A.1 in Appendix). Since 0-hop query has a larger overfitting level (Figure 8 in Appendix), it should have larger attack accuracy.

The combined attack takes the inputs of the 0/1/2-hop attack models as its input and the result is summarized in Figure 1a. We find that the combined attack achieves higher attack performance since it takes the advantage of 0/1/2-hop attacks. For instance, when the target model is GCN trained on Citeseer, the membership inference accuracy is 0.823 for the combined attack, while only 0.795/0.691/0.684 for the 0/1/2-hop attacks.




(a) Degree (b) Ego Density (c) Feature Similarity

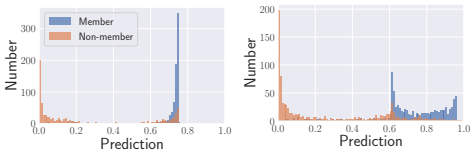
Figure 3: AUC for 0-hop, 1-hop, 2-hop, and combined attacks on different groups of nodes categorized by degree, ego density, and feature similarity on Cora-full. The architecture of both target and shadow model is GraphSAGE. The x-axis represents different groups, e.g, 0-25 for degree means the group of nodes whose degrees are in the lowest 25% of the dataset. The y-axis represents the AUC.

**Node Property.** We next investigate which kinds of member nodes are more prone to membership inference. To this end, we calculate three metrics for each member node, i.e., *degree*, *ego density*, and *feature similarity*. The first two are related to a node's graph property and the last one focuses on the node's feature. For a given node $v$, the degree of the node is defined as the number of edges connected to it. Ego density measures the graph density of a node $v$'s 2-hop subgraph $g^2(v)$. Feature similarity measures how similar a node $v$'s feature to nodes' features in its 2-hop subgraph $g^2(v)$. Specifically, we calculate the similarity

(cosine similarity) between the feature of $v$ and the feature of each node in $g^2(v)$. Then, we average all the similarity.

We categorize all the nodes in $\mathcal{D}_{Target}^{Train}$ into four different groups (as the positive cases) based on their degrees, ego density, and feature similarity, respectively. Nodes in $\mathcal{D}_{Target}^{Test}$ are considered as the negative cases for each group. We summarize the results when both target and shadow model are GraphSAGE trained on Cora-full in Figure 3 (see also Section A.4 for the results on other datasets). Note that the distribution of member and non-member nodes in each group is not uniform, thus we utilize AUC (area under the ROC curve) to measure the attack performance in each group as AUC is not sensitive to imbalanced classes (Backes et al., 2017; Fredrikson et al., 2014).

In general, we find that higher degree leads to lower AUC score for all attacks (see Figure 3a). For instance, the 0-hop attack's AUC is 0.853 on nodes in the lowest 25% degree group while the AUC is 0.764 in the highest 25% degree group. Recall that during the training process, each GNN layer generates a node's representation by aggregating its neighbor nodes' representation. With a higher degree, more neighbor nodes are involved, which may reduce the "exposure" of the target node itself, thus lesser membership inference risk. In Figure 3b, we find that larger ego density implies higher attack performance. For instance, for GraphSAGE trained on Cora-full, the 1-hop attack achieves 0.781 AUC on nodes with less than 0.25 ego density while the AUC increases to 0.810 for nodes with larger than 0.75 ego density. The reason behind this can be credited to the aggregation function of GNN models. Higher density enables a node to participate more times in the aggregation process during training, which results in the model memorizing more information about the node. Also, such observation is rooted in the social homophily theory (Easley & Kleinberg, 2010), i.e., nodes in higher density subgraphs are more likely to share similar features. This makes the aggregation output of a node become more similar to the node feature itself which makes the node easier to be memorized by the target model. We further measure the relation between attack performance and feature similarity (see Figure 3c). Our finding reveals that membership inference is indeed more effective when the target node has a larger feature similarity with its neighbors. For GraphSAGE trained on Cora-full, the 2-hop attack's AUC increases from 0.766 to 0.814 when the feature similarity increase from less than 0.25 to larger than 0.75.




(a) $\mathcal{A}_c$ (b) $\mathcal{A}_c^{cal}$

Figure 4: Prediction probability of being a member for the combined attack $\mathcal{A}_c$ and its calibrated version $\mathcal{A}_c^{cal}$ when target model is GraphSAGE trained on Citeseer.

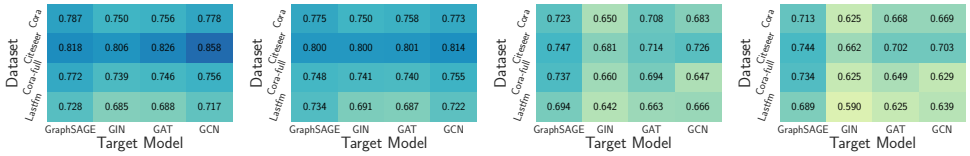
### 4.3 Calibrated Attacks

We summarize the performance of calibrated attacks in Figure 5. An observation is that the calibrated version of attacks in general achieves better performance. For instance, when the target model is GraphSAGE trained on Citeseer, the 1-hop attack $\mathcal{A}_1$ achieves 0.694 accuracy (Figure 1c) while its calibrated version reaches 0.747 accuracy (Figure 5c). To better illustrate this, take the GraphSAGE trained on Citeseer as an example, we summarize the probability of being a member for the combined attack $\mathcal{A}_c$ and its calibrated version $\mathcal{A}_c^{cal}$ in Figure 4. We observe that, when the attack model is trained with calibration, the overlap between members and non-members is smaller, which indicates that the attack model can better separate members from non-members.

### 4.4 Relax Assumptions

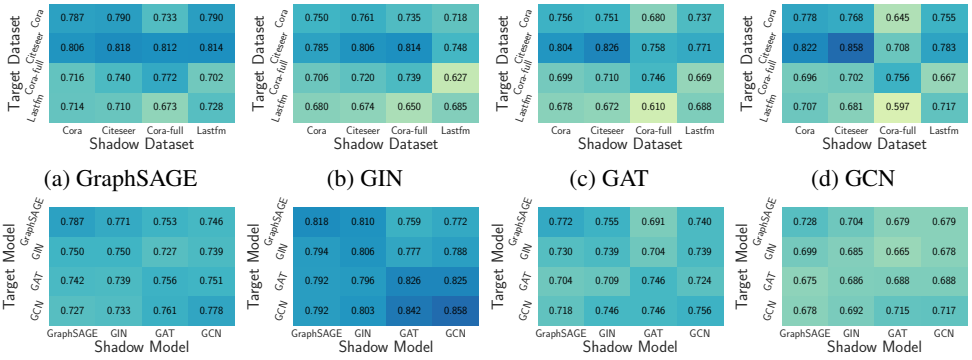
We further investigate whether the two key assumptions for our attacks (see Section 3.2) can be relaxed: 1) the adversary has a shadow dataset that comes from the same distribution as the target dataset, 2) the adversary has a shadow model with the same architecture as the target model. For instance, in Figure 7a, when the target model is GraphSAGE trained on Citeseer, the attack accuracy is 0.818 (0.806) with Citeseer (Cora) as the shadow dataset. This shows that even the adversary does not have the same distribution shadow dataset, they can still launch effective membership inference.

**Different Shadow Dataset Distribution.** The first row of Figure 7 shows the attack results when the shadow model is trained on a dataset from a different distribution. We observe that our calibrated




(a) Combined (b) 0-hop (c) 1-hop (d) 2-hop

Figure 5: The performance of calibrated attacks for combined, 0-hop, 1-hop, and 2-hop attacks for different GNN architectures on four different datasets.




(a) GraphSAGE (b) GIN (c) GAT (d) GCN

(e) Cora (f) Citeseer (g) Cora-full (h) Lastfm

Figure 7: Attack performance of $\mathcal{A}_c^{cal}$ when using: 1) different distribution shadow datasets to train the shadow models (the first row) and 2) different architectures to establish the shadow models (the second row).

version of combined attack can still achieve similar performance compared to the same distribution shadow dataset. To further investigate the reason behind, we extract the embeddings of members and non-members from two calibrated version of combined attack models (one corresponds to the same distribution shadow dataset, the other corresponds to the different distribution shadow dataset), and project the embeddings into a 2-dimensional space using t-SNE (van der Maaten & Hinton, 2008).

The results are shown in Figure 6. We observe that if we use different distribution shadow dataset to train the attack model, the embeddings of members and non-members might be distributed differently. For instance, the member nodes lie in the bottom left area in Figure 6a, while in the top left area in Figure 6b. However, for both of them, member and non-member nodes are easily separable, which indicate that the posteriors of members and non-members have different patterns and can be distinguished by the attack model.


(a) $\mathcal{D}_{Shadow}$: Cora (b) $\mathcal{D}_{Shadow}$: Citeseer

Figure 6: The embeddings of 100 randomly selected member and non-member nodes obtained from the combined attack's hidden layer. We project them into a 2-dimensional space using t-SNE. The target model is GraphSAGE trained on Cora. The two shadow models are GraphSAGE trained on Cora or Citeseer.

**Different Shadow Model Architecture.** The results for the attack using different shadow model architectures are summarized in the second row of Figure 7. We see that a shadow model with a different architecture from the target model still yields good attack performance. For instance, in Figure 7e, for GraphSAGE trained on Cora, the attack accuracy is 0.771 when the shadow model architecture is GIN while the original attack accuracy is 0.787. This indicates that the intrinsic difference between members and non-members is model agnostic and can be leveraged to perform effective attacks. Note that we also show the combined, 0-hop, 1-hop, and 2-hop attacks' performance in Section A.5 and the trends

are similar. In summary, both assumptions can be relaxed, which further demonstrates the severe membership privacy risks of GNNs.

### 4.5 Possible Defenses

To mitigate the membership inference attacks, we investigate three possible defense mechanisms, namely random edge addition, DP, and label-only output. We find that DP can better preserve the membership privacy while maintaining the performance on the original tasks. Section A.6 contains further details.

## 5 Discussion

We acknowledge that our attack methodology is similar to Olatunji et al. (2021) (which is the 2-hop attack mentioned in our paper). However, we empirically show that with this technology, an attacker can achieve better performance with limited information (0-hop and 1-hop attacks). Moreover, we propose a new calibrated attack that does not require multiple reference models, which is more query efficient. The results demonstrate that our proposed calibrated attack is more effective than the 0/1/2-hop attacks. In this paper, our attacks mainly focus on the node level. However, our calibrated attack can also be easily transferred into graph level to enhance the performance as well.

## 6 Related Work

Membership inference attacks aim at inferring membership of individual training samples of a target model to which an adversary has black-box access through a prediction API (Shokri et al., 2017; Salem et al., 2019; Nasr et al., 2018; Yeom et al., 2018; Hayes et al., 2019; Nasr et al., 2019; Chen et al., 2018; Song & Shokri, 2020; Carlini et al., 2019; Li & Zhang, 2021). Most of the existing attacks focus on deep learning models that are trained on sensitive data from non-structured data, such as images and texts. Shokri et al. (2017) propose the first membership inference attack against machine learning models in the black-box setting. Salem et al. (2019) further relax several key assumptions from (Shokri et al., 2017), such as knowledge of the target model architecture, shadow dataset from the same distribution. Yeom et al. (2018) discuss the relationship between overfitting and membership attacks. Nasr et al. (2019) conduct a comprehensive study for membership inference attacks in both black-box and white-box settings. Duddu et al. (2020) and Olatunji et al. (2021) have performed 2-hop attacks against GNNs. However, we empirically demonstrate that our proposed 1-hop and 0-hop attacks can perform better while requiring less information. We additionally propose combined attacks and the calibrated version of all attacks to further improve the attack performance.

## 7 Conclusion

In this paper, we perform a comprehensive privacy risk assessment of GNNs through the lens of node-level membership inference attacks. We systematically define the threat model along three dimensions, including shadow dataset, shadow model, and node topology, and propose eight different attack models. We conduct extensive experiments on four popular GNN models over four benchmark datasets. Our evaluation results show that GNNs are indeed vulnerable to membership inference attacks even with minimal background knowledge of an adversary. Also, our newly proposed calibrated attacks can further breach the membership privacy. Moreover, our analysis reveals that a node's degree, ego density, and feature similarity have a large impact to the attack performance. We further show that the attacks are still effective even the adversary does not have the same distribution shadow dataset or same architecture shadow model. To mitigate the attacks, we propose three defense mechanisms and discuss their trade-offs between membership privacy and model utility. We point out that the node-level membership inference attack can be launched in a more restricted scenario (i.e., 0-hop query) and we hope our work can inspire the community to develop stronger defenses based on our comprehensive analysis.

## REFERENCES

Michael Backes, Mathias Humbert, Jun Pang, and Yang Zhang. walk2friends: Inferring Social Links from Mobility Profiles. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1943–1957. ACM, 2017.

Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore Art Thou R3579X? Anonymized Social Networks, Hidden Patterns, and Structural Steganography. In *International Conference on World Wide Web (WWW)*, pp. 181–190. ACM, 2007.

Aleksandar Bojchevski and Stephan Günnemann. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *International Conference on Learning Representations (ICLR)*, 2018.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks. In *USENIX Security Symposium (USENIX Security)*, pp. 267–284. USENIX, 2019.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership Inference Attacks From First Principles. *CoRR abs/2112.03570*, 2021.

Qingrong Chen, Chong Xiang, Minhui Xue, Bo Li, Nikita Borisov, Dali Kaarfar, and Haojin Zhu. Differentially Private Data Generative Models. *CoRR abs/1812.02274*, 2018.

Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and Mobility: User Movement in Location-based Social Networks. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1082–1090. ACM, 2011.

Christopher A. Choquette Choo, Florian Tramèr, Nicholas Carlini, and Nicolas Papernot. Label-Only Membership Inference Attacks. In *International Conference on Machine Learning (ICML)*, pp. 1964–1974. PMLR, 2021.

David J. Crandall, Lars Backstrom, Dan Cosley, Siddharth Suri, Daniel Huttenlocher, and Jon Kleinberg. Inferring Social Ties from Geographic Coincidences. *Proceedings of the National Academy of Sciences*, 2010.

Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. Quantifying Privacy Leakage in Graph Embedding. *CoRR abs/2010.00906*, 2020.

David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

Matt 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 *USENIX Security Symposium (USENIX Security)*, pp. 17–32. USENIX, 2014.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 1322–1333. ACM, 2015.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 1025–1035. NIPS, 2017.

Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. LOGAN: Evaluating Privacy Leakage of Generative Models Using Generative Adversarial Networks. *Privacy Enhancing Technologies Symposium*, 2019.

Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing Links from Graph Neural Networks. In *USENIX Security Symposium (USENIX Security)*, pp. 2669–2686. USENIX, 2021.

Jinyuan Jia, Binghui Wang, Le Zhang, and Neil Zhenqiang Gong. AttriInfer: Inferring User Attributes in Online Social Networks Using Markov Random Fields. In *International Conference on World Wide Web (WWW)*, pp. 1561–1569. ACM, 2017.

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

Klas Leino and Matt Fredrikson. Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference. In *USENIX Security Symposium (USENIX Security)*, pp. 1605–1622. USENIX, 2020.

Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2014.

Zheng Li and Yang Zhang. Membership Leakage in Label-Only Exposures. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 880–895. ACM, 2021.

Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 497–512. IEEE, 2019.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine Learning with Membership Privacy using Adversarial Regularization. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 634–646. ACM, 2018.

Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 1021–1035. IEEE, 2019.

Iyiola E. Olatunji, Wolfgang Nejdl, and Megha Khosla. Membership Inference Attack on Graph Neural Networks. *CoRR abs/2101.06570*, 2021.

Benedek Rozemberczki and Rik Sarkar. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1325–1334. ACM, 2020.

Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2019.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *IEEE Symposium on Security and Privacy (S&P)*, pp. 3–18. IEEE, 2017.

Congzheng Song and Vitaly Shmatikov. Auditing Data Provenance in Text-Generation Models. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 196–206. ACM, 2019.

Congzheng Song and Vitaly Shmatikov. Overlearning Reveals Sensitive Attributes. In *International Conference on Learning Representations (ICLR)*, 2020.

Congzheng Song and Reza Shokri. Membership Encoding for Deep Learning. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*, pp. 344–356. ACM, 2020.

Liwei Song and Prateek Mittal. Systematic Evaluation of Privacy Risks of Machine Learning Models. In *USENIX Security Symposium (USENIX Security)*. USENIX, 2021.

Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 2008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Lauren Watson, Chuan Guo, Graham Cormode, and Alexandre Sablayrolles. On the Importance of Difficulty Calibration in Membership Inference Attacks. *CoRR abs/2111.08440*, 2021.

Fan Wu, Yunhui Long, Ce Zhang, and Bo Li. LinkTeller: Recovering Private Edges from Graph Neural Networks via Influence Analysis. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *IEEE Computer Security Foundations Symposium (CSF)*, pp. 268–282. IEEE, 2018.

Jiaxuan You, Jonathan M. Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-aware Graph Neural Networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, pp. 10737–10745. AAAI, 2021.

Table 1: Dataset statistics.

| Dataset | #. Node | #. Edge | #. Feature | #. Class |
|---|---|---|---|---|
| Cora | 2,708 | 5,429 | 1,433 | 7 |
| Citeseer | 3,327 | 4,732 | 3,703 | 6 |
| Cora-full | 19,793 | 65,311 | 8,710 | 70 |
| Lastfm | 7,624 | 27,806 | 7,842 | 18 |

# A Appendix

## A.1 Training Details of Target and Attack Models

**Target Models.** We leverage four GNN architectures, i.e., GraphSAGE, GAT, GIN, and GCN, to construct our target models and shadow models. For each target model, we set the number of layers to 2 and the number of neurons to 32 in the hidden layer. Additionally, GAT models require the specification of the number of heads in the multi-head attention mechanism. We set the number of heads for the first layer and the second layer to 2 and 1, respectively. We also use dropout in all hidden layers to reduce overfitting, and the dropout rate is 0.5. We adopt cross-entropy as the loss function and Adam as the optimizer. The learning rate is set to 0.003. The target and shadow models are both trained for 200 epochs.

**Baseline Model.** We leverage a 2-layer MLP as the baseline model to perform the same task as the target model's original task. Each hidden layer has 32 neurons with ReLU as its activation function. Loss function, optimizer, epochs, and learning rate are identical to those of the target GNN models.

**Attack Models.** For 0-hop, 1-hop, and 2-hop attacks, a 2-layer MLP is utilized as the attack model and the number of neurons in the hidden layer is set to 128. Regarding the combined attack, the inputs of 0-hop, 1-hop, and 2-hop attacks are first fed into three separated linear layers (with 64, 32, and 32 neurons) simultaneously. We then concatenate the three embeddings (128 dimensions in total) and feed them to another linear layer for membership inference. ReLU is adopted as the activation function for all the attack models. Also, the loss function and optimizer are the same as the target model. We set the learning rate to 0.001 and the training epochs to 500.

**Implementation.** Our code is currently implemented in Python 3.9.12 with PyTorch 1.10 and DGL 0.7.1, and run on an NVIDIA DGX-A100 server with Ubuntu 18.04.

## A.2 Target Model Performance

We show the performance of the target models with respect to their original classification tasks in Figure 8. To get the posteriors of a given node, we consider three query scenarios for each target model, i.e., 0-hop, 1-hop, and 2-hop query. For comparison, we only consider a node's feature as the input to each baseline model (i.e., a 2-layer MLP) and perform the same classification task as the target model.

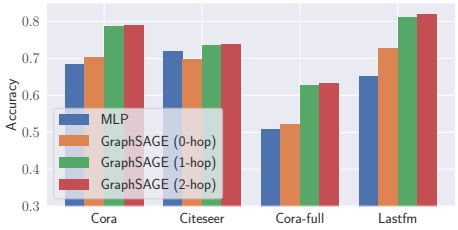



Figure 8: The performance of original classification tasks when the target model's architecture is MLP or GraphSAGE (0-hop, 1-hop and 2-hop query). The x-axis represents different datasets. The y-axis represents the original classification tasks' accuracy.

Due to space limitations, we only show the results for GraphSAGE. Other GNN models exhibit similar trends. First of all, compared to MLP, we observe that GNN has higher performance in the original task when using 2-hop queries. For instance, on the Cora dataset, the baseline MLP achieves 0.684 accuracy while the GraphSAGE (2-hop) achieves 0.790 accuracy. This demonstrates the efficacy of GNN models that consider nodes' features as well as their neighborhood information jointly for classification. Second and more interestingly, 1-hop or

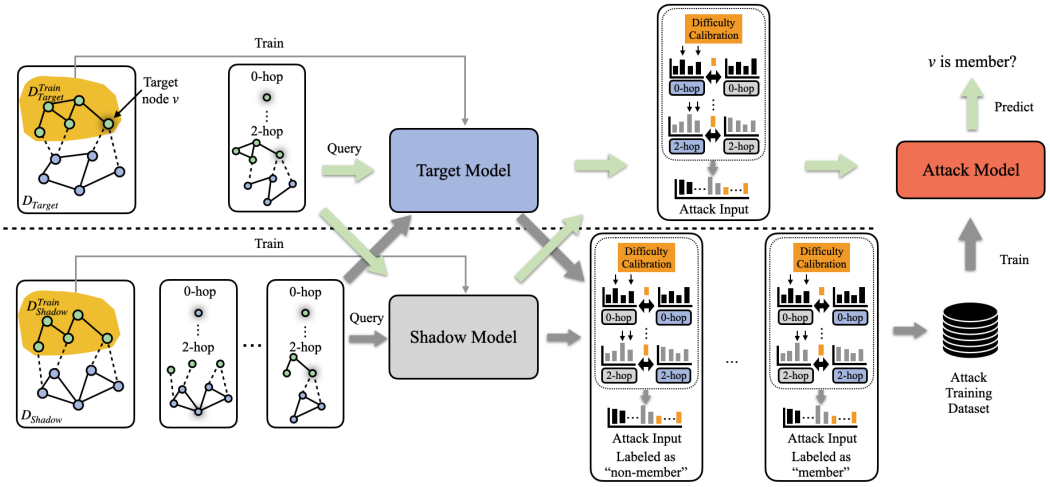
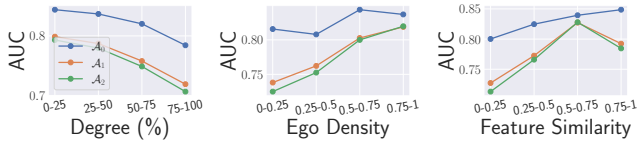



Figure 9: A schematic overview of node-level membership inference attack against GNNs. Note that for the combined attack, we conduct 0-hop, 1-hop, and 2-hop query to obtain the inputs of the attack model.




(a) Degree (b) Ego Density (c) Feature Similarity

Figure 10: AUC for 0-hop, 1-hop, 2-hop, and combined attacks on different groups of nodes categorized by degree, ego density, and feature similarity on Cora. The architecture of both target and shadow model is GraphSAGE. The x-axis represents different groups, e.g, 0-25 for degree means the group of nodes whose degrees are in the lowest 25% of the dataset. The y-axis represents the AUC.

even 0-hop query on GraphSAGE also achieves better performance than MLP in most of the cases. This indicates that the graph information used during the training phase can be generalized to boost the performance of a GNN model even when it is queried with only a node's feature (0-hop query) or incomplete neighborhood information (1-hop query).

### A.3 Attack Pipeline

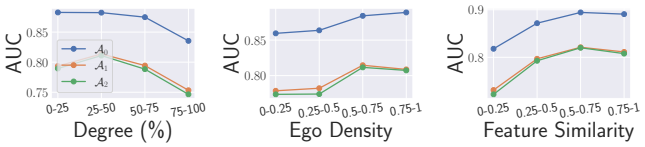
Figure 9 shows a schematic overview of node-level membership inference attack against GNNs.

### A.4 The Effect of Node Property on Attack Performance

Figure 10c, Figure 11c, and Figure 12c summarize the results on Cora, Citeseer, and Lastfm.

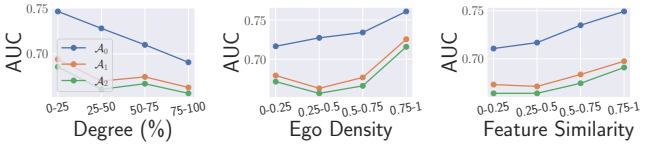
### A.5 Relax Assumptions

The combined, 0-hop, 1-hop, and 2-hop attack's performance when using different distribution shadow datasets to train the shadow models are shown in Figure 13, Figure 14, Figure 15, and Figure 16, respectively. The combined, 0-hop, 1-hop, and 2-hop attack's performance when using different architectures to establish the shadow models are shown in Figure 17, Figure 18, Figure 19, and Figure 20, respectively.



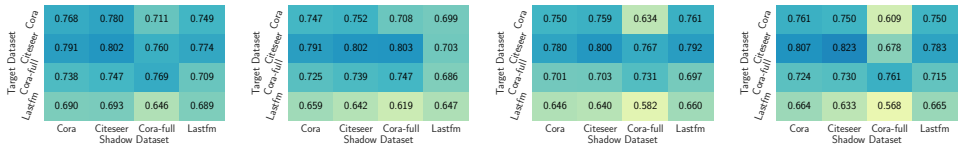

(a) Degree (b) Ego Density (c) Feature Similarity

Figure 11: AUC for 0-hop, 1-hop, 2-hop, and combined attacks on different groups of nodes categorized by degree, ego density, and feature similarity on Citeseer. The architecture of both target and shadow model is GraphSAGE. The x-axis represents different groups, e.g, 0-25 for degree means the group of nodes whose degrees are in the lowest 25% of the dataset. The y-axis represents the AUC.



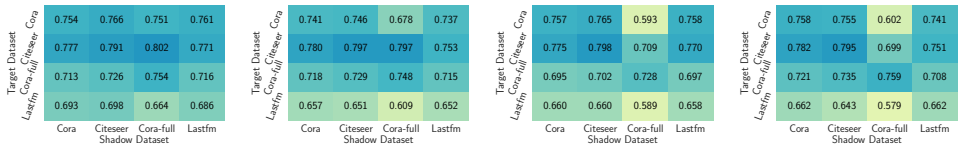

(a) Degree (b) Ego Density (c) Feature Similarity

Figure 12: AUC for 0-hop, 1-hop, 2-hop, and combined attacks on different groups of nodes categorized by degree, ego density, and feature similarity on Lastfm. The architecture of both target and shadow model is GraphSAGE. The x-axis represents different groups, e.g, 0-25 for degree means the group of nodes whose degrees are in the lowest 25% of the dataset. The y-axis represents the AUC.




(a) GraphSAGE (b) GIN (c) GAT (d) GCN

Figure 13: Attack performance of $\mathcal{A}_c$ when using different distribution shadow datasets to train the shadow models. The caption for each sub-figure denotes the target and shadow models' architectures.

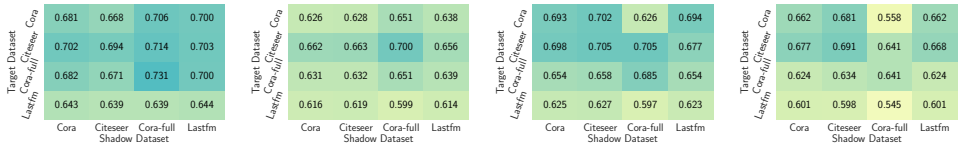



(a) GraphSAGE (b) GIN (c) GAT (d) GCN

Figure 14: Attack performance of $\mathcal{A}_0$ when using different distribution shadow datasets to train the shadow models. The caption for each sub-figure denotes the target and shadow models' architectures.

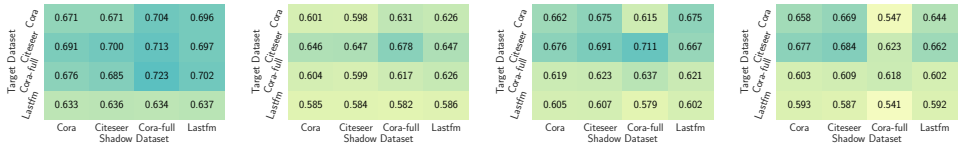
### A.6 Possible Defenses

**Random Edge Addition.** In the first defense, we perturb the target training dataset's graph structure by randomly adding edges. For the adversary, the shadow model is trained on the original shadow training dataset. We evaluate the target models' performance with respect to the original classification task, i.e., utility, and the membership inference attack performance using the combined attacks. Due to space limitations, we only show the results when both the target and shadow models use GraphSAGE as their architecture in Figure 21. Other models exhibit similar trends.



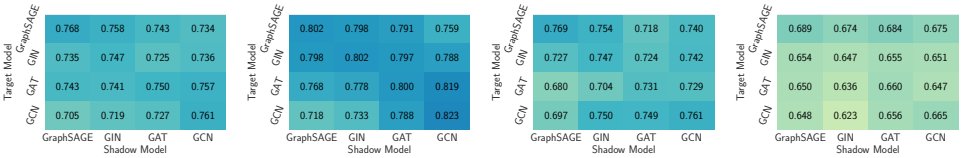

(a) GraphSAGE (b) GIN (c) GAT (d) GCN

Figure 15: Attack performance of $\mathcal{A}_1$ when using different distribution shadow datasets to train the shadow models. The caption for each sub-figure denotes the target and shadow models' architectures.



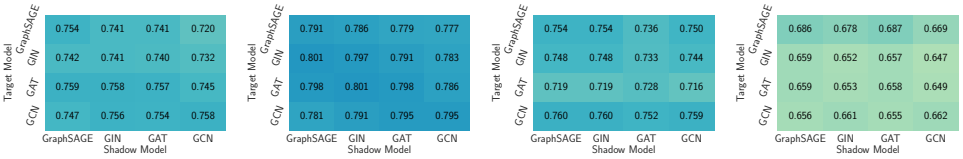

(a) GraphSAGE (b) GIN (c) GAT (d) GCN

Figure 16: Attack performance of $\mathcal{A}_2$ when using different distribution shadow datasets to train the shadow models. The caption for each sub-figure denotes the target and shadow models' architectures.



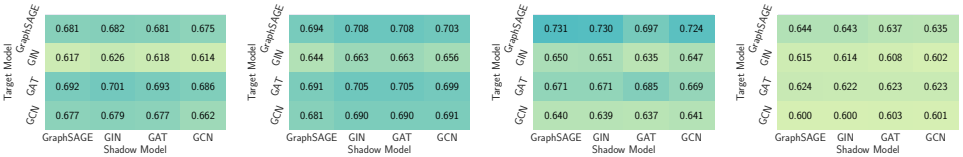

(a) Cora (b) Citeseer (c) Cora-full (d) Lastfm

Figure 17: Attack performance of $\mathcal{A}_c$ when using different architectures to establish the shadow models. The caption for each sub-figure denotes the target and shadow datasets.




(a) Cora (b) Citeseer (c) Cora-full (d) Lastfm

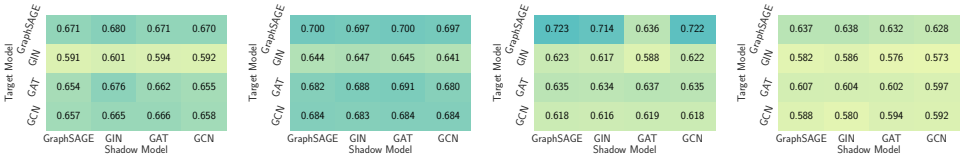
Figure 18: Attack performance of $\mathcal{A}_0$ when using different architectures to establish the shadow models. The caption for each sub-figure denotes the target and shadow datasets.




(a) Cora (b) Citeseer (c) Cora-full (d) Lastfm

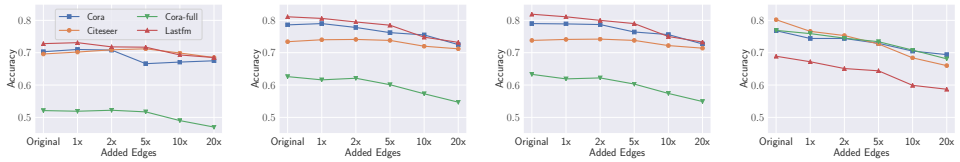
Figure 19: Attack performance of $\mathcal{A}_1$ when using different architectures to establish the shadow models. The caption for each sub-figure denotes the target and shadow datasets.

In Figure 21d, we observe that with more random edges added, the attack performance indeed drops. For instance, the membership inference accuracy is 0.802 on the original Citeseer dataset, while the



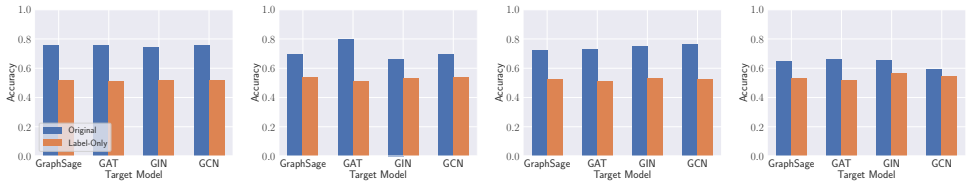

(a) Cora (b) Citeseer (c) Cora-full (d) Lastfm

Figure 20: Attack performance of $\mathcal{A}_2$ when using different architectures to establish the shadow models. The caption for each sub-figure denotes the target and shadow datasets.




(a) Target Performance (0-hop) (b) Target Performance (1-hop) (c) Target Performance (2-hop) (d) Attack Performance ($\mathcal{A}_c$)

Figure 21: The performance of the target model's original task and membership inference attacks when applying random edge addition as the defense. The x-axis represents different proportions of edges added. Here, $2\times$ means randomly adding in total 2 times more edges in the target training dataset. The y-axis represents the accuracy of the target models' original classification tasks or membership inference attacks. Note that we only show the results when GraphSAGE is used as the architecture for both target and shadow models.




(a) Cora (b) Citeseer (c) Cora-full (d) Lastfm

Figure 22: The performance of membership inference attacks when applying label-only output as the defense. The x-axis represents different target models' architectures. The y-axis represents the accuracy of membership inference attacks.

accuracy drops to 0.660 when 20 times more edges are added. This indicates that adding random edges to the target training dataset can protect nodes' membership privacy. As shown in Figure 10a, nodes with higher degree suffer less membership leakage risks, since the aggregation function of GNN during training aggregates more neighbor nodes' information and "memorize" less about the target node itself. On the other hand, the target models also suffer large utility loss as shown in Figure 21a and Figure 21c. For instance, the accuracy of the original classification task is 0.819 on the original Lastfm dataset using 2-hop query, while the accuracy decreases to 0.733 when 20 times more edges are added (the corresponding attack accuracy drops from 0.687 to 0.584).

**Label-Only Output.** For the second defense, we let the target model only return the prediction label instead of posteriors. In this case, we assume that the adversary knows the total number of classes of the target model. The adversary first converts the prediction labels derived from the 0-hop, 1-hop, and 2-hop queries into three one-hot vectors, respectively. Then, the three vectors serve as the input to train the combined attack model $\mathcal{A}_c$.

The performance of membership inference attacks against different target models is shown in Figure 22. We observe that on all the target models, membership inference attack accuracy decreases significantly, which is close to the random guess baseline (50%). For instance, on Cora-full, when both the target and shadow models' architectures are GraphSAGE, the membership inference accu-

Table 2: Utility and Attack performance against DP-GCN ($\epsilon = 8$).

| Dataset | Target Performance | | | |
|---|---|---|---|---|
| | GraphSage | GAT | GIN | GCN |
| Cora | 0.774 (-0.016) | 0.765 (-0.041) | 0.775 (-0.023) | 0.802 (-0.010) |
| Citeseer | 0.729 (-0.005) | 0.721 (-0.014) | 0.722 (+0.001) | 0.744 (+0.006) |
| Cora-full | 0.632 (-0.004) | 0.634 (+0.008) | 0.621 (+0.008) | 0.632 (+0.000) |
| Lastfm | 0.807 (-0.008) | 0.794 (-0.008) | 0.813 (-0.014) | 0.816 (-0.009) |
| Dataset | Attack Performance | | | |
| | GraphSage | GAT | GIN | GCN |
| Cora | 0.690 (-0.079) | 0.696 (-0.040) | 0.640 (-0.107) | 0.693 (-0.081) |
| Citeseer | 0.721 (-0.088) | 0.736 (-0.091) | 0.649 (-0.153) | 0.716 (-0.107) |
| Cora-full | 0.757 (-0.009) | 0.717 (-0.018) | 0.723 (-0.019) | 0.729 (-0.024) |
| Lastfm | 0.636 (-0.052) | 0.615 (-0.029) | 0.628 (-0.024) | 0.613 (-0.067) |

racy of the original combined attack is 0.769, while the accuracy drops to 0.523 after applying the label-only output defense. The defense is more effective since it decreases the attack performance while preserving the original task's performance. However, it may also limit the target model's utility as labels contain less information than posteriors. Also, in many cases, the ML tasks require posteriors to understand the model's prediction confidence. We note that previous work (Choo et al., 2021; Li & Zhang, 2021) investigates the label-only membership inference attack on non-GNN models. However, its effectiveness on the proposed defense for GNN models remains unjustified and we leave it as our future work.

**Differential Privacy (DP).** We then consider training the target model with differential privacy (DP). Specifically, we apply the LAPGRAPH algorithm (Wu et al., 2022) which first computes the number of edges that needs to be kept in the perturbed graph using a small portion of the privacy budget $\epsilon_1$. Then, the Laplace noise will be added to the whole adjacency matrix with the remaining privacy budget $\epsilon_2$ and keeps the largest number of edges in the perturbed graph. The total privacy budget $\epsilon = \epsilon_1 + \epsilon_2$. In our experiment, we set $\epsilon = 8$ for all models. The target model performance (with the 2-hop query) and the attack performance ($\mathcal{A}_c$) are summarized in Table 2. We observe that DP can achieve a better utility-privacy trade-off than random edge addition. For instance, for the GraphSage model with DP trained on Cora, the target model's performance is 0.774 and the attack performance is 0.690, which are both better than random edge addition (shown in Figure 21). This is because DP perturb edges in a more find-grained way than random perturbing.

In summary, our proposed defense mechanisms can achieve a trade-off between membership privacy and model utility. In the future, we plan to investigate more advanced defense mechanisms.

### A.7 Overfitting vs. Attack Performance

**Theorem A.1.** *Let $g = p - q$ be the overfitting level where $p$ and $q$ is the training and testing accuracy of the target model on its original classification tasks. If the total numbers of members and non-members are balance, with the posteriors and the ground truth label, a lower bound of membership inference attack accuracy is $\frac{1+g}{2}$.*

*Proof.* Given a sample, we predict it as a member if it is correctly classified by the target model otherwise a non-member. For all member nodes, $p$ of them are correctly classified and $1-p$ of them are wrongly classified. For all non-member nodes, $1-q$ of them are correctly classified and $q$ of them are wrongly classified. In that sense, the overall accuracy is $\frac{p+(1-q)}{1+1} = \frac{1+g}{2}$. □