Link Stealing Attacks Against Inductive Graph Neural Networks

Yixin Wu
CISPA Helmholtz Center for Information Security
yixin.wu@cispa.de

Xinlei He
Hong Kong University of Science and Technology
xinleihe@hkust-gz.edu.cn

Pascal Berrang
University of Birmingham
P.P.Berrang@bham.ac.uk

Mathias Humbert
University of Lausanne
mathias.humbert@unil.ch

Michael Backes
CISPA Helmholtz Center for Information Security
director@cispa.de

Neil Zhenqiang Gong
Duke University
neil.gong@duke.edu

Yang Zhang
CISPA Helmholtz Center for Information Security
zhang@cispa.de

ABSTRACT

A graph neural network (GNN) is a type of neural network that is specifically designed to process graph-structured data. Typically, GNNs can be implemented in two settings, including the transductive setting and the inductive setting. In the transductive setting, the trained model can only predict the labels of nodes that were observed at the training time. In the inductive setting, the trained model can be generalized to new nodes/graphs. Due to its flexibility, the inductive setting is the most popular GNN setting at the moment. Previous work has shown that transductive GNNs are vulnerable to a series of privacy attacks. However, a comprehensive privacy analysis of inductive GNN models is still missing. This paper fills the gap by conducting a systematic privacy analysis of inductive GNNs through the lens of link stealing attacks, one of the most popular attacks that are specifically designed for GNNs. We propose two types of link stealing attacks, i.e., posterior-only attacks and combined attacks. We define threat models of the posterior-only attacks with respect to node topology and the combined attacks by considering combinations of posteriors, node attributes, and graph features. Extensive evaluation on six real-world datasets demonstrates that inductive GNNs leak rich information that enables link stealing attacks with advantageous properties. Even attacks with no knowledge about graph structures can be effective. We also show that our attacks are robust to different node similarities and different graph features. As a counterpart, we investigate two possible defenses and discover they are ineffective against our attacks, which calls for more effective defenses.

KEYWORDS

Graph Neural Networks (GNNs), Link Stealing Attacks

1 INTRODUCTION

Many types of data can be represented in graphs, such as social networks, molecules, transportation, etc. Being modeled in a non-Euclidean space, the graph-structured data has imposed formidable challenges on traditional deep learning algorithms. In order to make full use of the graph-structured data, a graph version of neural networks has emerged [14, 18, 31, 40], which is known as graph neural networks (GNNs). In practice, GNNs have enabled several successful applications like drug discovery and online service recommendation. The core idea of GNNs is to leverage neighbor information among nodes to obtain node embeddings. These node embeddings can then be directly used to predict unknown node labels. There are mainly two settings for GNNs: the transductive setting and the inductive setting. In the transductive setting, all the data, including the training dataset and the testing dataset without labels, are used during training. The GNN model learns from seen data and predicts labels of the testing dataset. Since the data in the testing dataset is used in the training phase, transductive GNNs cannot be generalized to unseen nodes. However, in practice, there are a considerable amount of application scenarios where graphs dynamically evolve, e.g., the growth of social networks, thereby making the transductive GNN models less practical. Unlike GNN models in the transductive setting, inductive GNN models can predict labels of unseen nodes. Therefore, inductive GNNs are the most popular GNNs at the moment.

Graph-structured data often needs to remain private because data owners spend a large number of resources collecting it or because the data contains inherently sensitive information [4, 13]. For example, social relationships (encoded by edges in the graph) often represent privacy-sensitive information among users [33]. In another example, discovering drugs whose compounds' chemical structures are represented as graphs is extremely costly, making these chemical graph structures valuable assets [22].

Previous research has shown that GNN models are vulnerable to privacy attacks, e.g., to membership inference attacks that aim to infer whether a given node [16, 23] or graph [35] is used to train the target GNN model. More importantly, given that edge information

Proceedings on Privacy Enhancing Technologies 2024(4), 818–839
© 2024 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2024-0143

is very privacy-sensitive and is the most significant factor differentiating GNNs from other machine learning models, previous studies have focused on the problem of edge privacy. He et al. [15] propose link stealing attacks where different attack strategies are evaluated under eight threat models. However, they only concentrate on the transductive setting, a less realistic scenario.

In this work, we explore the possibility of link stealing attacks in the inductive setting. Unlike the transductive setting, where the adversary only needs to feed the nodes' IDs to obtain predictions based on full neighbor information, the inductive setting poses a different challenge. Here, the adversary relies on their own, often limited and incomplete, neighbor information to construct nodes' subgraphs before feeding them into the target model for prediction. Although the inductive setting makes the link stealing attacks more challenging, our evaluation shows that our attacks work well with limited or even no neighbor information. Wu et al. [36] propose a query-based attack strategy, LinkTeller, aiming to recover links among nodes of interest in the inductive setting. LinkTeller has two strong assumptions that are key to the success of the attack: (1) there is a model owner who is aware of the complete graph structure among nodes of interest and can obtain the complete neighbor information each time receiving a query; (2) the adversary is allowed to query the target model with the same set of nodes multiple times. In our work, we relax these assumptions by (1) only querying the target model with the adversary's own constructed subgraphs that have incomplete or even no neighbor information each time and (2) querying the target model once per node. Our work stands in a more common scenario where the adversary needs to design attack input, such as nodes' subgraphs, based on their background knowledge.

We propose two types of link stealing attacks that infer whether there exists a link between a given node pair in the target training graph. We refer to the graph used in the target GNN model training dataset as the target training graph. The first attack is the posterior-only attack, which relies on the posteriors obtained from a target black-box GNN model to design the attack input features. The threat model of the posterior-only attack distinguishes between different types of node topology. As GNNs commonly use two layers (which correspond to the node neighborhood), we consider three types of node topology for our posterior-only attacks, i.e., 0-hop, 1-hop, and 2-hop. In 0-hop, the adversary has no knowledge about the training graph structure. In 1-hop/2-hop, the adversary knows 1-hop/2-hop subgraphs for the given node pairs. Note that the subgraph information is incomplete, as we do not assume the adversary has knowledge about the edge they want to infer. Additionally, the adversary can have background knowledge, including a shadow dataset containing a graph with node attributes and labels.

We then propose the combined attack, which combines features used in the traditional link prediction methods with the posteriors to design attack input features. Previous work solves the link prediction problems using node attributes and graph features that are proven effective in evaluation [1, 12, 19]. Therefore, we define the threat model of the combined attack against GNNs along three dimensions: in addition to the posteriors, we also consider node attributes and graph features. For the graph features, we consider network proximity measures based on node neighborhoods [19]. For the posteriors, we again distinguish between the three types of node topology. Even if the adversary does not have access to the subgraphs, they can still obtain the 0-hop posteriors. The adversary can combine posteriors with either node attributes, graph features, or both. In total, we design seven combined attacks in which the adversary uses different combinations of these three features (posteriors, node attributes, and graph features).

We evaluate our three posterior-only attacks and seven combined attacks on six real-world datasets. Traditional link prediction methods that predict links between nodes in the target training graph are regarded as baseline attacks. First, the results show that when the adversary has no knowledge about the graph structures, our attacks can achieve outstanding performance, e.g., outperforming the baselines by 14.2% AUC on DBLP. This demonstrates that posteriors leak rich information, even with no knowledge about the graph structures, which facilitates link stealing from the inductive GNN models. Second, our combined attacks can achieve higher AUC scores than baselines on all datasets, which indicates the inductive GNN models leak extra information to enhance the link stealing attack. Our attacks outperform the baselines by an average of 6.5% AUC and up to 12.5% on DBLP. Third, we further find that our attacks are more robust to different graph features and different node attributes' cosine similarities (abbreviated as node similarities) than the traditional link prediction methods. This favorable property can help the adversary detect "surprising" links that the baseline attacks cannot detect. It demonstrates that posteriors can enable link stealing attacks with high robustness against different graph features and different node similarities and improve the robustness of traditional link prediction methods. In addition, experiments show that even if we relax the assumptions, such as using different distributions or different sizes of shadow datasets, different architectures of shadow models, and different architectures of attack models, the link stealing attacks are still effective. Given these advantageous properties, our attacks represent a severe threat against inductive GNN models in real-world scenarios. In order to mitigate these attacks, we further investigate a label-only defense mechanism and the state-of-the-art DP-GCN mechanisms [36]; the results indicate these defenses are ineffective against our attacks.

In a nutshell, we summarize the contributions as follows:

- We propose two types of link stealing attacks, i.e., posterior-only attacks and combined attacks against inductive GNNs.
- We define the threat models for the posterior-only attacks based on node topology and combined attacks along three dimensions, i.e., posteriors, node attributes, and graph features. In total, we propose ten link stealing attacks depending on the adversarial settings.
- We extensively evaluate our ten attacks on six real-world datasets. The results show that inductive GNN models leak rich information to enable link stealing attacks that are robust to different graph features and different node similarities. Moreover, our attacks are still effective in most cases after applying two well-established defense mechanisms.

2 PRELIMINARIES

2.1 Graph Neural Networks

In this paper, we focus on the inductive GNN models trained for node classification tasks that aim to determine nodes' labels. Given a node v, the GNN model first learns its node embedding and then leverages the embedding to classify the label of this node. To generate node embeddings, the GNN model aggregates information from local network neighborhoods using neural networks. Generally, every node in a k-layer GNN has a receptive field of a k-hop neighborhood that determines the embedding of the node. $N^k(v)$ represents a node v's neighbors that are k-hop away. The k-hop subgraph of node v is denoted as $g^k(v)$. Same as traditional neural networks, GNNs can be of arbitrary depth. However, as the number of GNN layers increases, the shared neighbors between nodes also quickly grow, and GNNs might suffer from the over-smoothing problem [5] when k is large. Thus, k is usually set to two at most.

Given a graph $G = (V, E)$ with input node attributes $\{x_v, \forall v \in V\}$, V represents the set of all nodes in graph G, and E denotes the set of all edges in the graph. To obtain final node embeddings of v, node v first computes "transformed" message of v's k-hop neighborhood $N^k(v)$. Then, node v aggregates these "transformed" messages from k-hop neighborhood $N^k(v)$. After that, node v conducts a non-linear transformation to update its representation. Formally, each layer of the GNN model can be defined as:

$$\begin{aligned} m_u^{(l)} &= \mathcal{M}^{(l)}(h_u^{(l-1)}, u \in \{N(v) \cup v\}), \\ z_v^{(l)} &= \mathcal{A}^{(l)}(\{m_u^{(l)}, u \in N(v)\}, m_v^{(l)}), \\ h_v^{(l)} &= \mathcal{U}(z_v^{(l)}), \end{aligned} \tag{1}$$

where $N(v)$ is the neighborhood of v, $m_u^{(l)}$ is the message of node u, $z_v^{(l)}$ denotes the hidden state of v in layer l, and $h_v^{(l)}$ represents the representation of node v in layer l. The GNN model first initiates all the node embeddings $h_u^{(0)}$ using node attributes x_u. The $\mathcal{M}(\cdot)$ function computes messages for nodes including node v and its neighbors. Usually, a linear layer is used as the $\mathcal{M}(\cdot)$ function structure. The $\mathcal{A}(\cdot)$ function then aggregates messages from neighbors. To prevent information of node v itself from getting lost, the $\mathcal{A}(\cdot)$ function also aggregates the message from node v itself. The $\mathcal{U}(\cdot)$ function usually conducts a non-linear operation to update the representation of node v.

In this paper, we focus on four different GNN architectures, i.e., GraphSAGE [14], Graph Convolutional Network (GCN) [18], Graph Attention Network (GAT) [31], and Graph Isomorphism Network (GIN) [40]. More details on these models can be found in Appendix A.

2.2 GNN Settings

As illustrated in Figure 1, there are two settings for GNNs: transductive setting and inductive setting. In the transductive setting, the GNN model learns the embeddings of nodes that are contained in the training graph G^{Train}. Note that in the transductive setting, the entire graph G with its node attributes X_V can be observed in both the training and testing phases, i.e., $G = G^{Train} = G^{Test}$. During the training phase, we use G with node attributes and only a small subset of labels. During the testing phase, the users feed the identifiers of unlabeled nodes in G to query the GNN model and obtain prediction results. It means that the users including the adversary can obtain the posteriors based on full neighbor information, even though they do not have any knowledge of the neighbor information or node attributes at the testing time.

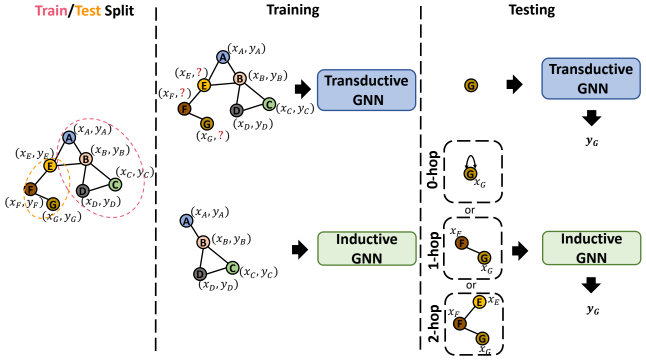


Figure 1: Comparison between the transductive and the inductive settings.

When there is a need for predicting unseen nodes, the inductive setting comes on stage. In the inductive setting, the GNN model learns aggregation functions that can induce the embeddings of unseen nodes. As illustrated in Figure 1, the training and testing datasets have independent graphs. The model owner first trains the GNN model using the training graph G^{Train} of the training dataset D^{Train}. Then, at the testing time, given a target node v in the testing graph G^{Test}, the users need to construct a k-hop subgraph $g^k_{(v)}$ containing node attributes and graph structure of $N(v)$ based on their knowledge and then feed it into the trained GNN model to predict the label of v. We refer to the query using k-hop subgraph $g^k_{(v)}$ as k-hop query, where $k \in \{0, 1, 2\}$. The classification ability can be generalized to previously unseen nodes, thus the inductive GNNs have broader application scenarios.

He et al. [15] have demonstrated that transductive GNNs are vulnerable to link stealing attacks. Here, the posteriors are predicted based on the full neighbor information including the link the adversary aims to infer. However, in the inductive setting, since the trained models have not seen the neighbor information of the testing data during the training phase, the user (including the adversary) can only obtain posteriors based on the subgraphs they constructed. Consequently, the neighbor information is always incomplete because of the link the adversary aims to infer, making the link stealing attacks against inductive GNNs significantly more challenging than those against transductive GNNs.

3 OUR ATTACKS

3.1 Problem Definition

Suppose there is an inductive target GNN model M_T trained on the target training dataset D^{Train}_{Target}. The model owner provides users with an inference API to obtain prediction results, which is often the case in practice. Like general users, the adversary can query M_T, but they utilize these prediction results for malicious purposes. Specifically, given a node pair (u, v) in the target training graph G^{Train}_{Target}, the adversary's goal is to infer whether there exists a link between nodes u and v.

Table 1: Attack input features for all attacks.

Attack	Attack	Posteriors			Node	Graph	Attack	Attack	Posteriors			Node	Graph
Method	Model	0-hop	1-hop	2-hop	Attribute	Feature	Method	Model	0-hop	1-hop	2-hop	Attribute	Feature
Attack-0	$A^0_{(p,\,\cdot,\,\cdot)}$	✓	✗	✗	✗	✗	Attack-7	$A^2_{(p,\,\cdot,\,g)}$	✗	✗	✓	✗	✓
Attack-1	$A^1_{(p,\,\cdot,\,\cdot)}$	✗	✓	✗	✗	✗	Attack-8	$A^1_{(p,\,n,\,g)}$	✗	✓	✗	✓	✓
Attack-2	$A^2_{(p,\,\cdot,\,\cdot)}$	✗	✗	✓	✗	✗	Attack-9	$A^2_{(p,\,n,\,g)}$	✗	✗	✓	✓	✓
Attack-3	$A^0_{(p,\,n,\,\cdot)}$	✓	✗	✗	✓	✗	Baseline-0	$B_{(\cdot,\,n,\,\cdot)}$	✗	✗	✗	✓	✗
Attack-4	$A^1_{(p,\,n,\,\cdot)}$	✗	✓	✗	✓	✗	Baseline-1	$B_{(\cdot,\,\cdot,\,g)}$	✗	✗	✗	✗	✓
Attack-5	$A^2_{(p,\,n,\,\cdot)}$	✗	✗	✓	✓	✗	Baseline-2	$B_{(\cdot,\,n,\,g)}$	✗	✗	✗	✓	✓
Attack-6	$A^1_{(p,\,\cdot,\,g)}$	✗	✓	✗	✗	✓							

3.2 Threat Model

Adversary's Background Knowledge. As mentioned above, we first assume that the adversary can only have black-box access to the target GNN model M_T, which is the most challenging setting [26, 27, 30]. To obtain the posteriors of v, the adversary needs to construct g_v^k that contains node attributes and graph structures. Node attributes can be easily acquired. For instance, the adversary can crawl social media content to extract node attributes for users in a social network. We discuss graph structures with different node topologies based on the adversary's knowledge in the following subsection. We also assume that the adversary can have a shadow dataset D_{Shadow} that contains node attributes, graphs, and ground-truth labels. Following previous work [30], we assume we can get D_{Shadow} that comes from the same distribution as D_{Target}. Note that D_{Target} and D_{Shadow} do not have any overlap in terms of nodes or edges. The adversary can train a shadow model M_S to mimic the behavior of M_T using D_{Shadow}. The later evaluation results show that the distribution/size of the shadow dataset and the architecture of the shadow model only has a slight effect on the attack performance.

Posterior-Only Attacks. This type of attack only uses the posteriors of the node pair (u, v) obtained from the GNN models to design attack input features. Different from the previous work [15, 36] where the target GNN models only take nodes of interest as input and leverage full neighbor information, we consider the most common scenario where the adversary needs to design attack input, i.e., nodes' subgraphs, based on their background knowledge. As the GNN models are in the inductive setting, we define the threat model of posterior-only attacks by categorizing the node topology in 0-hop, 1-hop, and 2-hop. We consider these three cases because there are usually at most two layers in inductive GNN models (see Section 2.1). In the 0-hop case, the adversary only knows the node attributes of u and v. In other words, the adversary has no neighbor information. As the input format of the GNN model is a graph, the adversary can add a self-loop edge for node u and v respectively to construct graphs and conduct a 0-hop query to get the node posteriors. In the 1-hop case, the adversary has knowledge about 1-hop subgraphs for u and v except for the link aiming to infer, meaning that the 1-hop subgraph is actually incomplete. The adversary can feed the incomplete 1-hop subgraphs into the target GNN model and perform a 1-hop query to get node posteriors for two nodes. The 2-hop case is similar to 1-hop except the adversary knows the incomplete 2-hop subgraphs of u and v (the link between u and v is missing).

Combined Attacks. Inspired by previous work [12, 19], we propose combined attacks that combine posteriors with traditional link prediction features, i.e., node attributes and graph features. We define threat models of combined attacks against GNNs along three dimensions, i.e., posteriors, node attributes, and graph features. Specifically, we consider three cases that combine posteriors with either of the other two features or both of them. In the first case, besides obtaining posteriors, the adversary can also leverage node attributes, as neighbor node attributes have an intrinsic similarity. Thus, the adversary can combine posteriors with node attributes to design attack input features. Also, the posteriors can be further divided into three types, i.e., 0-hop, 1-hop, and 2-hop. In the second case, the adversary can perform the 1-hop/2-hop query using 1-hop/2-hop subgraphs. They can further use subgraphs to generate graph features, e.g., common neighbors, which can reflect the connectivity between node neighborhoods. The natural intuition is that node pairs with higher connectivity are more likely to be linked in a graph structure. In the last case, the adversary makes full use of all background knowledge, i.e., knowledge of all three dimensions, to launch the link stealing attacks. Note that, in the latter two cases, the adversary can only generate graph features when performing 1-hop/2-hop queries, as they do not have any graph information in the 0-hop query case.

3.3 Attack Methodology

Shadow Model Training. The adversary first divides D_{Shadow} evenly into two disjoint splits. As it is an inductive setting, each split contains an independent graph with node attributes, edges, and labels. The first split is treated as the shadow training dataset D_{Shadow}^{Train} to train M_S, while the second half is considered as the shadow testing dataset D_{Shadow}^{Test} to evaluate M_S.

Attack Model Training. The attack model is a binary classifier that can predict if two given nodes have a connection in G_{Target}^{Train}. Therefore, the output of the attack model is 1/0, indicating whether there exists a link between the given pair or not. Its input is derived from the GNN's outputs (i.e., posteriors), node attributes, and network proximity measures based on node neighborhoods. To construct the attack training dataset D_{Attack}^{Train}, the adversary first needs to query M_S using subgraphs with node attributes from the

Table 2: Pairwise operations.

Operator	Definition
Hadamard	$f_i(u) * f_i(v)$
Average	$\frac{f_i(u)+f_i(v)}{2}$
Weighted-L1	$\lvert f_i(u) - f_i(v) \rvert$
Weighted-L2	$\lvert f_i(u) - f_i(v) \rvert^2$

shadow training dataset to get the posteriors. The attack input features of the posterior-only attack are only designed over posteriors. The adversary can conduct the 0-hop query, 1-hop query, or 2-hop query based on the background knowledge, which leads to different attack taxonomies of the posterior-only attacks. Depending on the background knowledge, the adversary can also leverage either node attributes or graph features, or both of them. Different types of features and query methods used to design attack input features together result in different taxonomies of the combined attacks. The attack models of different attacks are referred to as A^{β}_{α}, where A is short for the attack, β is the number of hops queried by the adversary, and $\alpha = (p,\ n,\ g)$ is a combination of characters: p, n, and g represent posteriors, node attributes, and graph features, respectively Attack input features for all attacks are shown in Table 1.

Taxonomy of Posterior-Only Attacks (Attack-0 to Attack-2). We present an overview of the proposed posterior-only attacks, combined attacks, and baselines in Figure 2. The posterior-only attacks leverage posteriors to design attack input features. For a given node pair, the adversary first feeds each node with its subgraph and node attributes into the GNN model to get the posteriors. As the attack input features should be the same regardless of the input order of nodes, we take a step further to process the posteriors of the given node pair. Specifically, we follow the strategy in [15] and leverage four pairwise, commutative operations to ensure attack input features are the same even if nodes are presented in a different order for the given pair. These pairwise operations are summarized in Table 2. In later Section 4, we perform a grid search to determine the exact pairwise operation for the proposed attacks. After feeding posteriors into these operations, we concatenate these results as the attack input feature for each pair. For posterior-only attacks (Attack-0/1/2), the attack models are defined as $A^{\beta}_{(p,\ \cdot,\ \cdot)}$. To launch these attacks, we adopt MLP models that take features generated from 0/1/2-hop posteriors as input.

Taxonomy of Combined Attacks (Attack-3 to Attack-9). In addition to using posteriors, the combined attack combines posteriors with either node attributes or graph features or both to design attack input features.

For the combined attacks that use posteriors and node attributes (Attack-3/4/5), the attack model is defined as $A^{\beta}_{(p,\ n,\ \cdot)}, \beta \in \{0, 1, 2\}$, as we assume the adversary has the knowledge to perform the 0/1/2-hop queries in this attack. Node attributes for node pairs also face the node order issue. Therefore, we apply the pairwise operations in Table 2 for node attributes as well. Due to its high dimensionality, we only apply the Hadamard operation for node attributes. The attack model $A^{\beta}_{(p,\ n,\ \cdot)}$ is a multiple-input MLP model that takes features generated by 0/1/2-hop posteriors and node attributes separately as input.

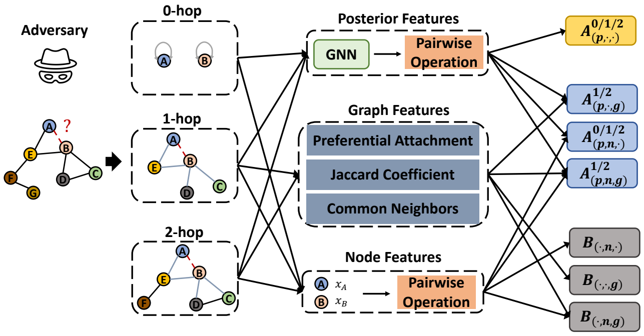


Figure 2: Overview of the proposed posterior-only attack, combined attack and baselines.

For the combined attacks that use posteriors and graph features (Attack-6/7), the attack model is defined as $A^{\beta}_{(p,\ \cdot,\ g)}, \beta \in \{1, 2\}$, as we assume the adversary has the knowledge to perform the 1-hop/2-hop queries in this attack. Note that there is no combination of 0-hop posteriors and graph features, as we assume the adversary has no knowledge about graph structures when they perform the 0-hop query. Following previous work [19], we focus on network proximity measures, including common neighbors, Jaccard coefficient, and preferential attachment, and concatenate these three features to be our graph features. The attack model $A^{\beta}_{(p,\ \cdot,\ g)}$ is a multiple-input MLP model that takes features generated from 1-hop/2-hop posteriors and graph features separately as input.

For the combined attacks that use posteriors, node attributes, and graph features (Attack-8/9), the attack model is defined as $A^{\beta}_{(p,\ n,\ g)}, \beta \in \{1, 2\}$. The attack model $A^{\beta}_{(p,\ n,\ g)}$ is a multiple-input MLP model that takes features generated from 1-hop/2-hop posteriors, node attributes, and graph features separately as input.

Baselines [1, 12, 19]. We compare the posterior-only attacks and combined attacks with three baselines. These baselines are used to perform traditional link prediction tasks that aim to infer whether there exists a link between a given node pair in the graph of interest. We can exploit these traditional link prediction methods to perform link stealing attacks, as the graph of interest is set to G^{Train}_{Target}. Baseline-0 only uses node attributes, and the attack model is referred to as $B_{(\cdot,\ n,\ \cdot)}$. Baseline-1 only uses graph features, and the attack model is referred to as $B_{(\cdot,\ \cdot,\ g)}$. Baseline-2 uses node attributes and graph features, and the attack model is referred to as $B_{(\cdot,\ n,\ g)}$. The attack model $B_{(\cdot,\ n,\cdot)}$ and $B_{(\cdot,\cdot,\ g)}$ are MLP models that take features generated from either node attributes or graph features as input. The attack model $B_{(\cdot,\ n,\ g)}$ is a multiple-input MLP model that takes features generated from node attributes and graph features separately as input.

Link Stealing. To enable the link stealing attack for a given node pair, the adversary first performs a 0-hop, 1-hop, or 2-hop query to obtain posteriors based on the background knowledge of node

Table 3: Dataset statistics.

Dataset	# Nodes	# Edges	Density	# Classes
Cora	2,995	8,416	0.00215	7
Pubmed	19,717	44,324	0.00028	3
DBLP	17,716	52,867	0.00039	4
Photo	7,650	143,663	0.00420	8
CS	13,752	287,209	0.00267	10
LastFM	7,624	63,236	0.00109	18

topology. Then, the adversary leverages posteriors alone or combines them with node attributes and graph features to generate attack input features. Finally, these input features are fed into the attack model to determine whether the given node pair has a link in G^{Train}_{Target}.

4 EVALUATION

In this section, we first introduce the details of the experimental setup. Second, we present the target performance and the attack performance and investigate to what extent our attacks can be a practical threat against inductive GNN models via relaxing assumptions. Third, we investigate the robustness of our attacks against different graph features and different node similarities. At last, we evaluate the performance of two defense mechanisms.

4.1 Experimental Setup

Datasets. We evaluate our attacks on six public datasets including Cora-ML (abbreviated as Cora) [2], Pubmed [28], DBLP [24], the AmazonCoBuy dataset for photos (abbreviated as Photo), the AmazonCoBuy dataset for computers (abbreviated as CS) [21], and LastFM Asia Social Network (abbreviated as LastFM) [25]. Among them, Cora, Pubmed, and DBLP are citation networks where nodes are publications and edges are citation links among these publications. The two Amazon datasets are part of the Amazon co-purchase graph where nodes are items, and edges between two items indicate that they have been purchased together. The LastFM dataset contains a social network where nodes are LastFM users from Asian countries, and edges are mutual follower relationships between them. General statistics, e.g., the number of edges and density, for all datasets are summarized in Table 3. Among all datasets, Pubmed and DBLP are two datasets with sparser density.

Dataset Configurations. We first randomly split the whole graph in half according to the number of nodes for each dataset. The first half is treated as the target dataset D_{Target}, while the second half constructs the shadow dataset D_{Shadow}. These two parts are disjoint. We discard the edges between nodes that end up in different splits; this affects the degree distribution of the resulting graphs at a minimal level. We further divide D_{Target} into the target training dataset D^{Train}_{Target} and target testing dataset D^{Test}_{Target} in an 8:2 ratio. D^{Train}_{Target} is used to train the target model M_T, while D^{Test}_{Target} can evaluate M_T's performance on the original task. The same processing procedure is applied to the shadow dataset D_{Shadow}. Note that each dataset contains an independent graph, node attributes, and node labels. For instance, G^{Train}_{Target}, along with its node attributes and node labels, is included in D^{Train}_{Target}. We use D_{Shadow} to generate the attack training dataset D^{Train}_{Attack}, while D_{Target} is leveraged to construct the attack testing dataset D^{Test}_{Attack}. The attack dataset contains input feature vectors of node pairs and labels indicating whether two nodes are linked or not. We refer to node pairs that are linked as positive pairs and node pairs that are not linked as negative pairs. To build D^{Train}_{Attack}, we first select all node pairs that are actually connected in the shadow training graph G^{Train}_{Shadow} as positive pairs. Following the negative sampling approach in previous work [15], we then randomly sample the same number of node pairs that are not connected in G^{Train}_{Shadow} as negative pairs. We apply the same processing procedure on the D^{Train}_{Target} to construct the D^{Test}_{Attack} that can evaluate all of our attacks.

Metrics. Following previous work [1, 15], we rely on the AUC score (area under the ROC curve) to evaluate the attack performance of link stealing attacks. We also leverage accuracy to evaluate the target model's performance.

Implementation Details. To train the models, we either follow the experimental setup of previous related work [15, 16, 29, 36] or conduct a grid search to find the best set of hyper-parameters. Below, we describe the final hyper-parameter set and architectures of the target, shadow, baseline, and attack models. We also report the search space of the hyper-parameters in Appendix B.

Target models. We leverage four GNN architectures, i.e., GraphSAGE, GCN, GAT, and GIN, to build the target models and shadow models. We follow previous work [15, 16, 36] and use a two-layer architecture with a full-neighbor sampler for each target model. We set the number of neurons to 128 in the hidden layer. The unit size of the output layer is determined by the number of classes of the original task. Each layer employs a ReLU activation and 0.5 dropout rate to reduce overfitting. In addition, the first layer of the GAT model has two attention heads, and the second layer only has one attention head. All models use cross-entropy as the loss function and Adam as the optimizer. The initial learning rate is set to 0.001. Both the target models and shadow models are trained for 200 epochs.

Baseline models. We use a three-layer MLP for Baseline-0. The hidden unit size of the first layer and the second layer are set to 128 and 32, respectively. The number of neurons in the output layer is set to two because the link stealing attack is a binary classification task. As the dimension of graph features is much smaller than that of node attributes, we only use an MLP model with two layers for Baseline-1. The hidden unit size is set to 16. Regarding Baseline-2, the two inputs are first fed into two sub-networks simultaneously. The sub-network for node attributes consists of three linear layers with 256 neurons, 64 neurons, and 8 neurons, respectively. Another sub-network is composed of one linear layer with one neuron. We concatenate the 8-dimensional embedding and 1-dimensional embedding together and feed them into a linear layer for link stealing. These baseline models also have the same activation function and drop rate in each layer. In addition, the loss function, optimizer, initial learning rate, and training epochs are the same as the target model. We use a cosine annealing scheduler to tune the learning rate in the training process.

Table 4: Test accuracy of the original tasks for different GNN architectures on six different datasets.

	Target Model				Baseline
Dataset	GraphSAGE	GIN	GAT	GCN	MLP
Cora	0.773	0.757	0.737	0.763	0.747
Pubmed	0.871	0.855	0.865	0.857	0.850
DBLP	0.739	0.727	0.746	0.746	0.726
Photo	0.935	0.877	0.859	0.855	0.790
CS	0.850	0.778	0.820	0.805	0.774
LastFM	0.752	0.738	0.721	0.760	0.718

Posterior-only attack models. We utilize a three-layer MLP as an attack model for all posterior-only attacks. These three linear layers have 128, 32, and 2 neurons, respectively. The loss function, optimizer, initial learning rate with its scheduler, and training epochs are the same as the baseline models.

Combined attack models. The attack model $A^{\beta}_{(p,\,n,\,\cdot)}$ is composed of two sub-networks and a linear layer. The first sub-network receives features generated from node attributes and comprises three linear layers with 128 neurons, 64 neurons, and 16 neurons, respectively. The second sub-network, consisting of two layers with 64 and 16 neurons, receives features derived from posteriors. These two 16-dimensional embeddings are concatenated and fed into a linear layer to make predictions. The attack model $A^{\beta}_{(p,\,\cdot,\,g)}$ also has two sub-networks and a linear layer. However, the first sub-network receives graph features and consists of two linear layers with 16 neurons and 4 neurons, respectively. The sub-network that receives posteriors is composed of three layers with 128 neurons, 64 neurons, and 16 neurons, respectively. The 4-dimensional and 16-dimensional embeddings are concatenated and fed into a linear layer to make predictions. The attack model $A^{\beta}_{(p,\,n,\,g)}$ consists of three sub-networks and a linear layer. The first sub-network that receives node attributes is composed of three layers with 128 neurons, 64 neurons, and 16 neurons, respectively. The second sub-network is the same as the first one, but it is used to receive posteriors. The third sub-network comprises only one linear layer with four neurons and receives graph features. These three embeddings are concatenated and fed into a linear layer to make predictions. The loss function, optimizer, initial learning rate with its scheduler, and training epochs of the combined attacks are the same as the baseline models. We show that, even if the attack model has different architectures, i.e., different number of layers, we can still get similar attack performance (see Appendix D).

Our codes[1] are mainly implemented with PyTorch,[2] Networkx,[3] and DGL.[4] All experiments are carried out on an NVIDIA DGX Server.

4.2 Target Model Performance

We first present in Table 4 the accuracy of the original node classification tasks for four GNN architectures on six different datasets. We can observe that all GNN models can achieve great target performance on all datasets. Following previous studies [16, 34], we construct a 2-layer MLP model as the baseline to undertake the same target classification tasks. We observe that our target GNN models consistently outperform the baseline. For instance, on the Photo dataset, the baseline MLP achieves an accuracy of 0.790, while GraphSAGE, GIN, GAT, and GCN achieve accuracies of 0.935, 0.877, 0.859, and 0.855, respectively. This demonstrates that employing graph neural networks to jointly use the node attributes and graph structures can well improve the node classification performance. We can also observe that the choice of architecture has a more significant impact on dense datasets. It is easily understood that these datasets have more complex neighbor relationships; thus, different aggregating methods can result in a larger variance in the target model performance. For instance, we consider the gap between the best and worst target performance on each dataset. It is 1.6% on the Pubmed dataset, while 8.0% on the Photo dataset. Compared to the Photo dataset, the Pubmed dataset is much sparser.

4.3 Our Attacks

We show the link stealing attack performance of the posterior-only attacks, combined attacks, and baseline attacks in Table 5. Due to the space limitation, we only show the results when both M_T and M_S are GraphSAGE. We observe that our attacks with 0-hop posteriors perform well. When the adversary has no knowledge about the graph structures, they can only perform Baseline-0, Attack-0, and Attack-3. We find that Attack-0 and Attack-3 outperform Baseline-0 on most of the datasets. For instance, on DBLP, Attack-3/Attack-0 outperforms Baseline-0 by 14.2%/8.9% AUC. It indicates that the inductive GNN can leak enough information to enable high-performing link stealing attacks, even only feeding into node attributes with self-loop edges. In other words, our attack is still effective even with no knowledge about the graph structures.

When the adversary has both node attributes and graph structures, they can perform all posterior-only attacks, combined attacks, and baseline attacks. Specifically, on the Cora, DBLP, and LastFM datasets, the posterior-only attacks are able to outperform all baseline attacks. The adversary can rely on the combined attacks to outperform baselines on the other three datasets. We also find that our attacks using the 1-hop query can achieve a performance similar to the 2-hop query. Compared to Attack-1, Attack-2 can only achieve an improvement of 0.15% AUC on average on all datasets. We can also get the same conclusion on the comparison of other 1-hop attacks and 2-hop attacks, e.g., Attack-8 and Attack-9. This demonstrates that inductive GNNs leak rich information even though the adversary only has limited knowledge about the graph structures (e.g., 1-hop subgraph). Overall, our attacks outperform the baselines by an average of 7.23% AUC on all six datasets, demonstrating that inductive GNNs are indeed more vulnerable to the proposed link stealing attacks than the baseline attacks. The posteriors are the most useful information, as they enable well-performing attacks and enhance the traditional link prediction methods. With more information, the combined attacks always outperform the posterior-only attacks. However, the attack model $A^{*}_{(p,\,\cdot,\,g)}$ surpasses the $A^{*}_{(p,\,n,\,g)}$ on the Photo and CS datasets. We postulate this is because the node attributes usually have high dimensions, which may lead to the

[1] https://github.com/yxoh/link_steal_pets2024.
[2] https://pytorch.org/.
[3] https://networkx.org/.
[4] https://www.dgl.ai/.

Table 5: Attack performance on all six datasets. The average AUC score of five runs is reported. Both the target model and the shadow model are GraphSAGE. The best performance of the posterior-only attacks and the combined attacks are highlighted in bold (A: Attack, B: Baseline).

	Baseline [1, 12, 19]			Posterior-Only Attack			Combined Attack						
Dataset	B0	B1	B2	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9
Cora	0.748	0.820	0.769	0.859	0.849	0.849	0.876	0.876	0.875	0.882	0.884	0.908	**0.909**
Pubmed	0.876	0.820	0.889	0.768	0.806	0.809	0.889	0.895	0.897	0.881	0.882	**0.939**	**0.939**
DBLP	0.692	0.787	0.804	0.781	0.821	0.822	0.834	0.873	0.872	0.879	0.903	0.924	**0.929**
Photo	0.813	0.930	0.878	0.877	0.898	0.898	0.892	0.916	0.915	0.967	**0.968**	0.946	0.946
CS	0.821	0.932	0.863	0.817	0.838	0.845	0.869	0.890	0.893	0.955	**0.956**	0.941	0.940
LastFM	0.798	0.786	0.866	0.850	0.869	0.867	0.883	0.909	0.911	0.919	0.921	0.929	**0.930**

Table 6: Attack performance of posterior-only attacks on six datasets when the target model is GraphSAGE and the shadow model is one of four architectures we mentioned in Section 2. The average AUC score of five runs is reported. The best performance of the posterior-only attacks is highlighted in bold.

		Architecture						Architecture			
Dataset	Method	GraphSAGE	GIN	GAT	GCN	Dataset	Method	GraphSAGE	GIN	GAT	GCN
Cora	Attack-0	0.859	0.853	0.858	**0.860**	Pubmed	Attack-0	**0.768**	0.764	0.763	0.765
	Attack-1	0.849	0.846	**0.850**	0.842		Attack-1	**0.806**	0.805	0.800	0.801
	Attack-2	0.849	0.843	**0.852**	0.843		Attack-2	0.809	**0.810**	0.805	0.806
DBLP	Attack-0	0.781	0.779	**0.782**	0.779	Photo	Attack-0	**0.877**	0.722	0.783	0.750
	Attack-1	0.821	0.820	0.818	**0.824**		Attack-1	**0.898**	0.785	0.801	0.810
	Attack-2	0.822	0.824	0.821	**0.829**		Attack-2	**0.898**	0.786	0.821	0.828
CS	Attack-0	**0.817**	0.770	0.798	0.785	LastFM	Attack-0	**0.850**	0.804	0.836	0.794
	Attack-1	0.838	**0.840**	0.837	0.823		Attack-1	0.869	0.855	**0.870**	0.833
	Attack-2	**0.845**	0.837	0.839	0.810		Attack-2	0.867	0.867	**0.871**	0.831

model's overfitting. In the following Section 4.4, we demonstrate that besides enabling high AUC attack performance, the posteriors also provide high robustness for link stealing attacks. We attribute these two favorable properties to the aggregation function of the GNN, as it capitalizes fully on the neighbor node attributes and graph structures. Due to space limitations, we present the attack performance on the other three target models in Appendix C, and the results demonstrate that our attacks are still effective. We also explore different layers of attack models in Appendix D, showing that the attacks are in general effective.

Different Shadow Models' Architectures. We then investigate the variants of different architectures of shadow models. We present the attack performance when the target model is GraphSAGE, and the shadow model's architecture is one of the four architectures mentioned in Section 2. As shown in Table 6, we only present the results of the posterior-only attacks on all six datasets since the different architectures only affect the posteriors. We find those shadow models with different architectures still yield excellent attack performance. We can also observe that the choice of the shadow model's architecture has a more significant impact on datasets with larger densities such as Photo and CS. As mentioned in Section 4.1, these datasets have more complex neighbor relationships, so different aggregating methods can lead to a larger variance in the attack performance of the posterior-only attacks. We calculate the gap between the best and worst attack performance of each posterior-only attack and take the average of all posterior-only attacks' gaps on each dataset. The gap is negligible on sparse datasets like DBLP and Pubmed. For example, the average gap is 0.55% on the DBLP and Pubmed datasets. The gap becomes larger on dense datasets such as CS and Photo. For instance, the average gap is 12.67% on the Photo dataset. Since node attributes and graph features do not depend on the shadow model's architecture, attackers can additionally rely on the combined attacks when they have no knowledge about the target model's architecture. We report the results of the combined attacks in Appendix E. As we can see, the combined attacks are architecture-agnostic. For example, the average gap is only 3.2% AUC using the combined attacks on the Photo dataset, and the gap is only 1.5% in Attack-5. Overall, shadow models with architecture different from the target model still achieve excellent attack performance. Hence, we can conclude that it is not necessary for the adversary to have knowledge of the target model's architecture as the choice of the shadow model's architecture does not have a significant impact on the attack performance.

Different Shadow Datasets' Distributions. In previous experiments, we leverage the shadow datasets from the same distribution as the target dataset to train the shadow models. Here, we relax the assumption by leveraging different distribution shadow datasets. To avoid the dimension mismatch problem, we redesign input features over the variable-length posteriors. Specifically, we calculate the entropy with pairwise functions for posteriors pairs and combine the

Table 7: Effect of the different distributions of the shadow datasets on the attack performance. Both the target and shadow architectures are GraphSAGE. The evaluation metric is the average AUC score of five runs.

Target	Shadow Dataset					
Dataset	Cora	Pubmed	DBLP	Photo	CS	LastFM
Cora	0.854	0.847	**0.867**	0.856	0.862	0.849
Pubmed	0.737	**0.814**	0.756	0.768	0.763	0.752
DBLP	0.782	0.805	**0.837**	0.788	0.794	0.785
Photo	0.863	**0.881**	0.879	0.876	0.874	0.872
CS	0.845	0.834	0.834	0.840	**0.841**	0.842
LastFM	0.881	0.882	0.876	0.882	0.880	**0.882**

Table 8: Effect of the shadow datasets' sizes. Both the target and shadow architectures are GraphSAGE. The target and shadow datasets are the same. The average AUC score of five runs is reported.

	Dataset Size				
Dataset	100%	50%	30%	20%	10%
Cora	**0.849**	0.833	0.839	0.827	0.789
Pubmed	**0.798**	0.776	0.770	0.762	0.742
DBLP	**0.823**	0.819	0.816	0.818	0.761
Photo	**0.902**	0.895	0.901	0.890	0.869
CS	**0.840**	0.839	0.825	0.832	0.823
LastFM	0.869	0.878	0.870	**0.884**	0.727

result vectors with cosine similarity, JS divergence, and correlation distance. These metrics measure the similarity between posteriors pairs. As shown in Table 7, the results on the diagonal are from datasets with the same distribution but with different input features, demonstrating that our newly designed input features are effective. We focus on Attack-1, as it only uses limited knowledge about graph structures but achieves a high attack success rate. We can observe that, with shadow datasets from dissimilar distributions, the posterior-only attacks can still achieve similar results on most datasets, even slightly better. When the target model is trained on sparse datasets like Pubmed and DBLP, the attack performance is more susceptible to the distribution of the shadow dataset. Overall, it demonstrates that information leaked from the GNN models can be transferred across datasets with dissimilar distributions.

Different Shadow Datasets' Sizes. So far, we assume the size of the shadow dataset and the target dataset are the same. However, it is likely that the attacker has no knowledge about the size of the target dataset. Therefore, we investigate the influence of shadow datasets' sizes on attack performance. Following the above experiment, we focus on Attack-1. We sample nodes in different proportions {10%, 20%, 30%, 50%, 100%} from the original shadow dataset to construct new shadow datasets. The results are summarized in Table 8. We can observe that the attack performance is almost the same when the dataset size changes from 20% to 100%. Even with only 10% dataset, the attack performance is still acceptable. Note that the proportion is divided by the number of nodes, and the corresponding number of edges is much smaller than the proportion. For example, the subgraph that contains 10% nodes only has 2.88% edges on the LastFM dataset, indicating our attacks are still effective with limited access to the same distribution shadow dataset. The reason is that the shadow model can still learn the "correct" predictions with a smaller dataset; hence the adversary can obtain the accurate similarities of the posteriors between node pairs. In conclusion, our attacks can be carried out effectively using a much smaller shadow dataset than the target dataset, which makes the attack more practical.

Takeaways. First, we show that when the adversary has no knowledge about the graph structures, our attacks can achieve outstanding performance, e.g., outperforming the baselines by 14.2% AUC on DBLP. It demonstrates that posteriors leak rich information, even with no knowledge about the graph structures, to steal links from the inductive GNN models. Also, our attacks still work well even with limited neighborhood information (i.e., attacks with 1-hop query perform similarly as those with 2-hop query). Second, when the adversary has both node attributes and graph structures, the combined attacks can achieve higher AUC scores than baselines on all datasets, which indicates the inductive GNN models leak extra information to enhance the link stealing attack. Our attacks outperform the baselines by an average of 6.5% AUC (on DBLP, it is 12.5% higher). Third, the experiments show that we can relax the assumptions about the shadow models' architectures, shadow datasets' distributions, and shadow datasets' sizes, as they do not have a significant impact on the attack performance. This advantageous property makes our attacks a serious threat against inductive GNN models in real-world scenarios.

4.4 Fine-Grained Analysis

Robustness. Previous work [19] shows that graph features based on node neighborhoods contribute to good link prediction performance, which is in line with the intuition that a node pair is more likely to be linked if their neighbors have a large overlap. Also, due to the intrinsic similarity of neighbor node attributes, traditional link prediction methods that use only node attributes can also be successful [12].

Such graph features and node attributes are also involved in our combined link stealing attacks. Thus, we select three graph metrics and a node metric and investigate the correlation between these metrics and our attack performance. Specifically, we investigate the robustness of the proposed attacks and baselines with respect to three graph metrics (common neighbors, preferential attachment, and Jaccard coefficient) and one node metric (node similarity). We categorize all existing links in the G_{Target}^{Train} (the positive pairs in D_{Attack}^{Test}) into ten groups based on these four metrics, respectively. We then calculate the AUC score for each group by combining node pairs in the group with all negative pairs in D_{Attack}^{Test}, as the AUC score is insensitive to imbalanced classification. As illustrated in Figure 3, we compare our attacks and baselines in terms of AUC scores on different groups sorted by four metrics on the LastFM dataset. We observe that the AUC score for Baseline-1 drops rapidly when the graph metric value decreases (group index increases), i.e., lower graph features result in worse attack performance. We also find that on decreasing the values of node similarity, Baseline-0/2

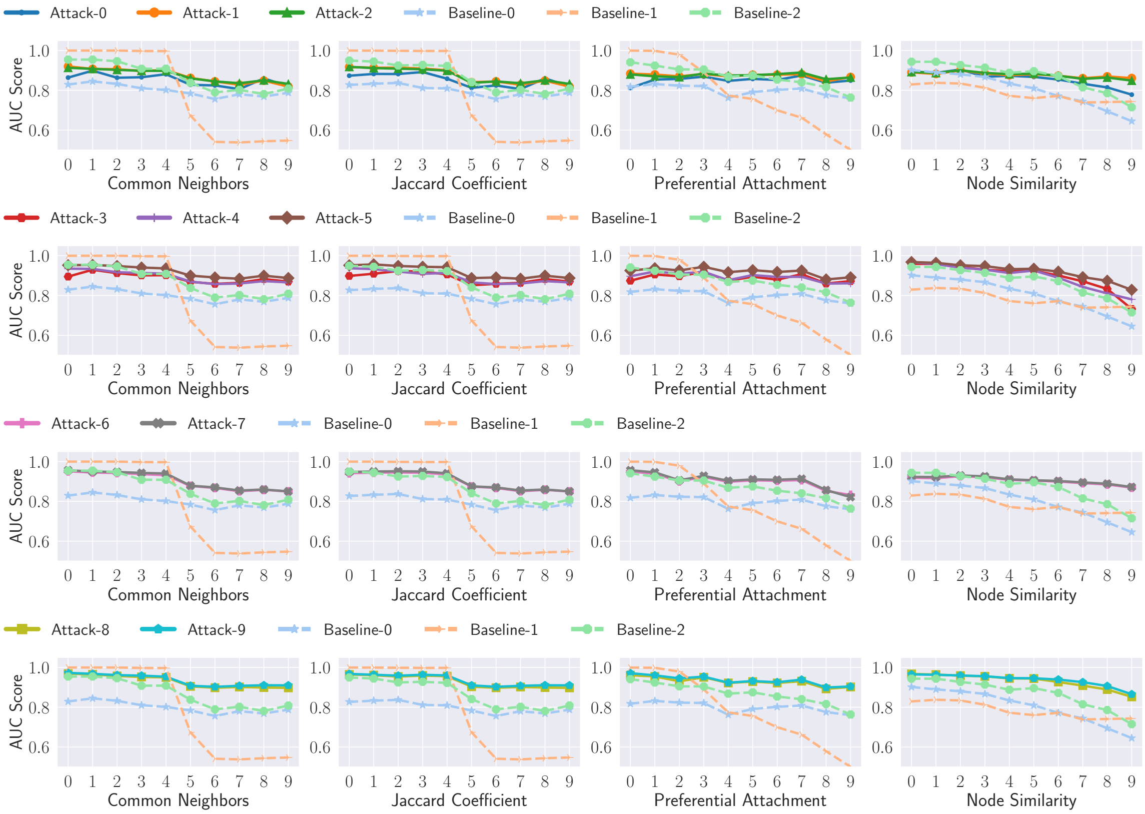


Figure 3: AUC score for our attacks and baselines on ten different groups on the LastFM dataset. The ten groups are formed by categorizing all existing links in G^{Train}_{Target} (the positive pairs in D^{Test}_{Attack}) based on node attributes and three graph metrics, respectively. The x-axis represents different groups in descending order of their corresponding metric values. The y-axis represents the AUC scores. Each column represents one metric. The first to fourth rows are $A^*_{(p,\cdot,\cdot)}$, $A^*_{(p,n,\cdot)}$, $A^*_{(p,\cdot,g)}$, and $A^*_{(p,n,g)}$, respectively. We report results on the other five datasets in Appendix F, and similar conclusions can be drawn.

has a significant deterioration in the attack performance, especially in the latter groups. It indicates that it is hard for the baseline attacks to distinguish between positive and negative pairs with either lower graph features' values or low node similarity, as the attack model tends to classify all of them as negative pairs. Although Baseline-0 and Baseline-2 do not show a substantial decreasing trend in the groups sorted by graph metrics, their AUC scores are relatively low in all ten groups. A similar situation exists for Baseline-1: as node similarity decreases, Baseline-1 is not significantly affected, but its overall attack performance is relatively low.

As for posterior-only attacks $A^*_{(p,\cdot,\cdot)}$, they have a stable performance on all groups sorted by either graph metrics or node similarity, showing the posteriors itself can enable robust link stealing attacks. As for combined attacks $A^*_{(p,n,\cdot)}$, $A^*_{(p,\cdot,g)}$, and $A^*_{(p,n,g)}$, they have high robustness with respect to different graph features and different node similarities and outperform baseline attacks by a large margin, showing that posteriors can help distinguish positive pairs with lower metric values. In other words, combining the posteriors, inherent graph features, and node attributes can result in more robust and better attack performance, especially in groups with lower metric values. Overall, these observations demonstrate that information leaks from inductive GNNs, i.e., posteriors, can improve both the attack performance and the robustness of link stealing attacks with respect to different graph features and different node similarities. We also show the comparison between our attacks and baselines on the other five datasets in Appendix F, and similar conclusions as above can be drawn. In addition, the analysis of robustness using Pearson correlation coefficient is also in Appendix F.

"Surprising" Links. In previous experiments, we demonstrate that our attacks achieve high robustness and accuracy. This conclusion drives us to explore if our attacks could detect "surprising" links, i.e., those links that are correctly predicted by our attacks but not by the baseline attacks. Specifically, we focus on Attack-1, the most efficient posterior-only attack, as we are interested in the difference between posteriors, node attributes, and graph features. We again categorize all positive pairs evenly into ten groups of positive pairs by four metrics, respectively. We only focus on the last groups where the metric values are the lowest. In Figure 4, each bar group presents the proportion of "surprising" links in the last group on different datasets. Different colors denote the last group

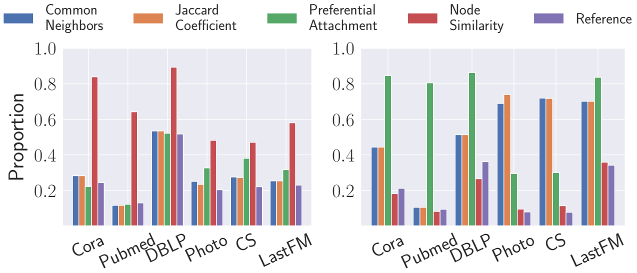


Figure 4: The proportion of the "surprising" links in the last group that is detected by Attack-1 but not by Baseline-0 (left figure) and by Attack-1 but not by Baseline-1 (right figure).

with respect to different metrics. We also use the proportion of "surprising" links in all positive pairs as a reference (purple). We observe that in the left figure, the proportions in the last group on all datasets with respect to node similarity are higher than the reference, indicating that links with lower node similarity can be better detected by Attack-1 but not by Baseline-0. We postulate this is because such "surprising" links are used by the aggregation function in the GNN training procedure, which may strengthen the memorization of such links. We have a similar observation in the right figure, i.e., posterior-only attacks are more robust than Baseline-1 against three different graph metrics.

Takeaways. In a nutshell, we find that our attacks are more robust with respect to different graph features and different node similarities than the traditional link prediction methods. This favorable property of our attacks can help the adversary detect "surprising" links that cannot be detected by the baseline attacks.

4.5 Possible Defenses

We investigate two defense mechanisms to mitigate link stealing attacks against inductive GNNs.

Label-Only Outputs. In this mechanism, the adversary can still perform queries with 0-hop, 1-hop, or 2-hop. However, the target model only returns prediction labels rather than posteriors. We assume the adversary has knowledge about the number of classes of the target model. The adversary can also perform multiple queries to get enough labels to know the number of classes. To avoid the node order issue, we convert two labels of a given node pair into two one-hot vectors and add them together as the attack input feature. As we can see in Figure 5, our attack performance has decreased after applying the label-only defense, especially on the DBLP and Pubmed datasets. The decrease in attack performance on DBLP and Pubmed is at most 5.8%. In addition, there is only a slight fluctuation on the other four datasets, which may be due to network homophily that states that nodes that have similar attributes/labels are more likely to be linked together. As illustrated in Appendix H, we also observe that the leading probability of the target model's outputs dominates the rest, especially in Attack-1 and Attack-2. This indicates the posteriors reveal almost the same amount of information as the labels, thus explaining why the link stealing attacks are still effective. We next explore whether our attacks still work when the leading probability is tight with the second-largest probability. Specifically, we leverage the "softmax temperature" technique [17] to make the posteriors softer. The temperature parameter T is set to 20. As illustrated in Figure 5, the attack performance increases rather than decreases on most datasets. We attribute this observation to the fact that the soft posteriors provide more information in different classes that can boost the link stealing attacks. Overall, our attacks can maintain good performance while dealing with different types of outputs.

DP-GCN. Another way to preserve the link privacy is to perturb the graph before training the GNN models. Concretely, we leverage two differentially private graph convolutional network (DP-GCN) mechanisms [36], i.e., EdgeRand and LapGraph. DP-GCN guarantees inference results should be indistinguishable on any pair of neighboring input graphs differing in one edge by perturbing the adjacency matrix. Specifically, the EdgeRand mechanism randomly flips cells in the adjacency matrix based on a Bernoulli random variable. The LapGraph mechanism first computes the number of edges T that needs to be kept in the perturbed graph using a small portion of the privacy budget and Laplace mechanism. Then, it adds the Laplace noise to the entire adjacency matrix using the remaining privacy budget and keeps the largest T entries in the perturbed graph.

We present target performance and attack performance on six datasets under privacy budget $\epsilon \in [1, 10]$, $\epsilon \in Z$ for two DP mechanisms, i.e., EdgeRand and LapGraph. Following previous work [36], both the target model and shadow model are GCN. We focus on Attack-1, as it only uses limited knowledge about graph structures but achieves a high success rate. The target performance, i.e., model utility, is reported in Figure 6a using the test accuracy as the metric, and the attack performance is reported in Figure 6b using the test AUC score as the metric.

For EdgeRand, we can observe that the attack performance and model utility decrease on all datasets via curtailing the privacy budgets since the privacy protection of the graph structure becomes stronger. When the attack performance is close to a random guess, the model also becomes no longer usable. It means that the attack success rate must be reduced at the cost of reducing target performance. It is expected that the perturbed graph may create some "fake" links to mitigate the attacks, but such links may also affect the target model's training, which leads to lower utility. Worse yet, the attack performance on the DBLP dataset does not drop significantly even at the cost of target performance.

For LapGraph, we can only see a sharp drop in the attack performance on two Amazon datasets, i.e., the Photo dataset and the CS dataset. The attack performance decreases along with the deterioration of model utility. The model utility decreases significantly compared with the vanilla GCN on the other four datasets, while the attack performance has barely dropped. Also, the model utility and attack performance do not decrease much with tightening privacy budgets. It may be possible that as we continue to make the privacy budget tighter, the attack performance decreases. Nevertheless, the target models are no longer usable when $\epsilon = 1$.

Takeaways. To mitigate these attacks, we investigate a label-only mechanism and two state-of-the-art DP-GCN mechanisms, i.e., EdgeRand and LapGraph. The results show that these defense mechanisms either have a negligible effect on the attack performance or decrease the attack performance while suffering unacceptable model utility reduction, indicating these defenses are impotent

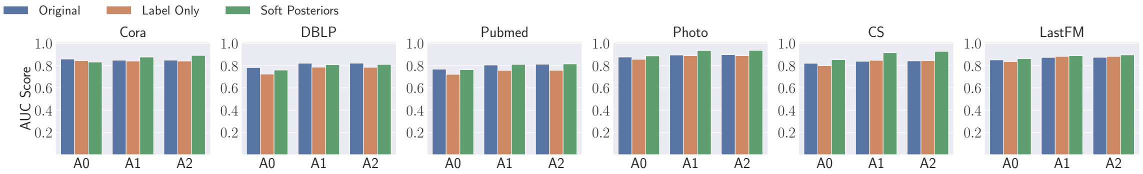
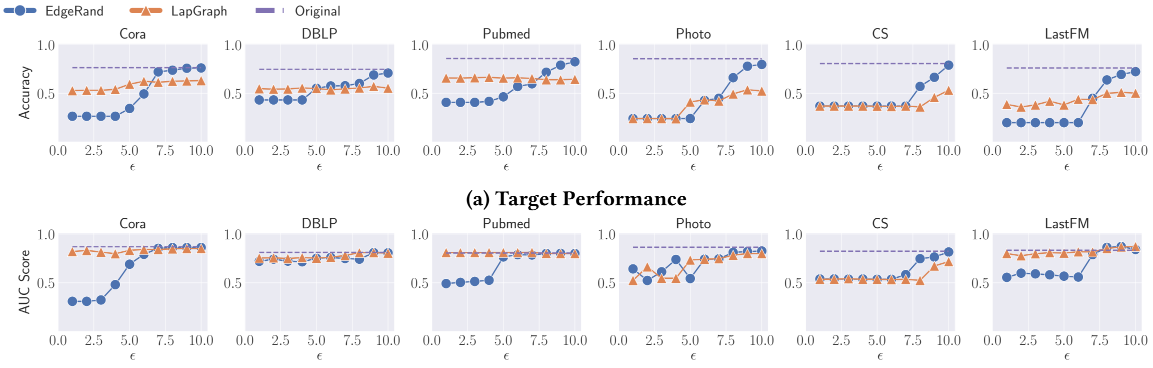


Figure 5: Attack performance of posterior-only attacks under three scenarios. We define different scenarios depending on the different outputs of the target model: original outputs generated by the softmax function (original), argmax outputs that only return the classification labels (label only), and soft outputs generated by softmax function with temperature scaling (soft posteriors). Each bar group corresponds to a posterior-only attack. Each bar within a group represents a scenario. The evaluation metric is the AUC score.



(a) Target Performance

(b) Attack Performance

Figure 6: (a) Target performance and (b) attack performance under varying privacy budgets $\epsilon \in [1, 10]$, when applying EdgeRand and LapGraph as defense mechanisms. Each column corresponds to a dataset. The x-axis denotes the privacy budget. The y-axis represents (a) target performance measured by accuracy and (b) attack performance of Attack-1 measured by AUC score. The dashed lines represent the (a) original target performance and (b) attack performance against vanilla GCN, while the solid lines represent the performance against two types of DP-GCN mechanisms.

against our attacks. We leave it as our future work to investigate more effective defenses.

5 RELATED WORK

Privacy Attacks Against GNNs. This topic has gained momentum over the last few years. Wu et al. [35] focus on graph-level membership inference attacks against GNNs trained for graph classification tasks. Olatunji et al. [23] propose a membership inference attack against node classification GNNs where the adversary's goal is to infer whether a specific node is used to train the target GNN model. He et al. [16] provide a complete attack taxonomy of node-level membership inference attacks by proposing three attacks where the adversary can perform a 0-hop, 1-hop, or 2-hop query based on different background knowledge. Duddu et al. [10] propose a membership inference attack in a white-box setting, where the adversary can leverage the intermediate outputs of the graph convolution layer, i.e., graph embeddings. Besides membership inference, Zhang et al. [43] also investigate privacy leakage of graph embeddings by mounting three inference attacks, i.e., property inference, subgraph inference, and graph reconstruction attacks.

There are two studies most relevant to our work. He et al. [15] propose the first link stealing attack that aims to infer whether there exists a link between a given node pair in the target model's training graph. Their work concentrates on the transductive setting, where the trained model has seen all testing data during the training phase, hence the users can only feed the identifiers of nodes to obtain prediction results. It means that the posteriors derived from transductive GNNs utilize full neighbor information including the link the adversary intends to infer, regardless of whether the adversary possesses this neighbor information at inference time. Our work shares similarities with [15] in using posteriors to design attack features. However, we focus on the inductive setting where the trained model has not seen the testing data during the training phase (see details in Section 2.2), and thus the adversary can only rely on limited and incomplete neighbor information, as the information of the link they intend to infer is missing, to construct nodes' subgraphs and then feed them into the target models to obtain posteriors. The absence of complete neighbor information makes the link stealing attacks much more challenging. Wu et al. [36] propose a query-based attack strategy, LinkTeller, which can recover edges among nodes of interest. It focuses on a specific scenario where different data holders can host edge information and node attributes separately. LinkTeller has two strong assumptions that are the key to the success of this work: (1) There is a model owner who is aware of the complete graph structure among nodes of interest and can obtain the complete neighbor information each time receiving a query; (2) The adversary is able to query the target model with the same set of nodes for multiple times. In our work, we concentrate on the privacy issue of the target model's training graph by inferring whether there exists a link between two given nodes. we also relax these assumptions in LinkTeller by (1) querying the target model with the adversary's own constructed subgraphs that have incomplete or even no neighbor information each time and (2) querying the target model once per node. Our work stands in a more common scenario where the adversary needs to design attack input, such as nodes' subgraphs, based on their background knowledge. There are some other works related to link stealing attacks, but they focus on different methodologies or properties. Ding et al. [9] propose a graph poisoning attack to increase the effectiveness of link stealing attacks. Zhang et al. [41] demonstrate that intra-class and inter-class node pairs have different levels of vulnerability to link stealing attacks.

Other Attacks Against GNNs. Recent studies have also shown that GNNs are vulnerable to other attacks such as model stealing attacks [8, 29, 34], backdoor attacks [38, 42] and adversarial attacks [3, 6, 7, 11, 20, 32, 37, 39, 42, 44, 45]. Regarding the model stealing attacks, DeFazio et al. [8] propose a model extraction attack that can steal GNN models in the transductive setting. Wu et al. [34] develop a series of model extraction attacks by categorizing the adversary's background knowledge along three dimensions, namely graph structures, node attributes, and shadow subgraphs. He et al. [29] propose model stealing attacks against inductive GNNs with six different attack scenarios. The adversary's background knowledge is categorized into two dimensions, i.e., the target model's response and query graph. Regarding the backdoor attacks, Zhang et al. [42] propose the first backdoor attack against GNNs for graph classification tasks, which can be enabled by injecting predefined subgraphs. Xi et al. [38] propose a backdoor attack against GNNs that the adversary can have no knowledge about the downstream tasks. Regarding the adversarial attacks, Bojchevski et al. [3] focus on poisoning attacks on unsupervised node embeddings. Chen et al. [6] propose two adversarial attacks for graph-based clustering. Dai et al. [7] propose graph adversarial attacks in three settings, namely white-box, practical black-box, and restricted black-box settings. Wang and Gong [32] propose adversarial attacks on collective classification methods via manipulating graph structures. Wu et al. [37] develop adversarial attacks by introducing the integrated gradient. Zhang et al. [42] propose the first subgraph-based backdoor attack on graph classification methods. Zügner et al. [44] propose an adversarial attack that can manipulate the node attributes and graph structures and preserve essential attributes of the graph at the same time.

6 CONCLUSION

In this paper, we comprehensively investigate the privacy leakage of the inductive graph neural networks through the lens of link stealing attacks. Specifically, we propose two types of attacks, i.e., posterior-only attack and combined attack. The threat model of the posterior-only attack is defined by categorizing node topology, while that of the combined attack is defined along three dimensions, i.e., posteriors, node attributes, and graph features. We conduct extensive experiments on four popular GNN models over six real-world datasets. The evaluation shows that inductive GNN leaks extra information to enable well-performing link stealing attacks even with weak assumptions. Moreover, our investigations reveal that our attacks are robust with respect to different graph features and different node similarities. To mitigate these attacks, we utilize label-only defense and the state-of-the-art DP-GCN mechanisms and show these defenses are impotent against our attacks.

ACKNOWLEDGMENTS

We thank all anonymous reviewers for their constructive comments. This work is partially funded by the European Health and Digital Executive Agency (HADEA) within the project "Understanding the individual host response against Hepatitis D Virus to develop a personalized approach for the management of hepatitis D" (D-Solve) (grant agreement number 101057917).

REFERENCES

[1] M. Backes, M. Humbert, J. Pang, and Y. Zhang, "walk2friends: Inferring Social Links from Mobility Profiles," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2017, pp. 1943–1957.

[2] A. Bojchevski and S. Günnemann, "Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking," in *International Conference on Learning Representations (ICLR)*, 2018.

[3] A. Bojchevski and S. Günnemann, "Adversarial Attacks on Node Embeddings via Graph Poisoning," in *International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 695–704.

[4] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective Data-Sanitization for Preventing Sensitive Information Inference Attacks in Social Networks," *IEEE Transactions on Dependable and Secure Computing*, 2018.

[5] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View," in *AAAI Conference on Artificial Intelligence (AAAI)*. AAAI, 2020, pp. 3438–3445.

[6] Y. Chen, Y. Nadji, A. Kountouras, F. Monrose, R. Perdisci, M. Antonakakis, and N. Vasiloglou, "Practical Attacks Against Graph-based Clustering," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2017, pp. 1125–1142.

[7] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial Attack on Graph Structured Data," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 1123–1132.

[8] D. DeFazio and A. Ramesh, "Adversarial Model Extraction on Graph Neural Networks," *CoRR abs/1912.07721*, 2019.

[9] R. Ding, S. Duan, X. Xu, and Y. Fei, "VertexSerum: Poisoning Graph Neural Networks for Link Inference," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2023, pp. 4509–4518.

[10] V. Duddu, A. Boutet, and V. Shejwalkar, "Quantifying Privacy Leakage in Graph Embedding," *CoRR abs/2010.00906*, 2020.

[11] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis, "All You Need Is Low (Rank): Defending Against Adversarial Attacks on Graphs," in *ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 2020, pp. 169–177.

[12] N. Z. Gong, A. Talwalkar, L. W. Mackey, L. Huang, E. C. R. Shin, E. Stefanov, E. Shi, and D. Song, "Joint Link Prediction and Attribute Inference Using a Social-Attribute Network," *ACM Transactions on Intelligent Systems and Technology*, 2014.

[13] A. Grover and J. Leskovec, "node2vec: Scalable Feature Learning for Networks," in *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2016, pp. 855–864.

[14] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Annual Conference on Neural Information Processing Systems (NIPS)*. NIPS, 2017, pp. 1025–1035.

[15] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang, "Stealing Links from Graph Neural Networks," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2021, pp. 2669–2686.

[16] X. He, R. Wen, Y. Wu, M. Backes, Y. Shen, and Y. Zhang, "Node-Level Membership Inference Attacks Against Graph Neural Networks," *CoRR abs/2102.05429*, 2021.

[17] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," *CoRR abs/1503.02531*, 2015.

[18] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations (ICLR)*, 2017.
[19] D. Liben-Nowell and J. Kleinberg, "The Link-prediction Problem for Social Networks," *Journal of the American Society for Information Science and Technology*, 2007.
[20] J. Ma, S. Ding, and Q. Mei, "Towards More Practical Adversarial Attacks on Graph Neural Networks," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.
[21] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-Based Recommendations on Styles and Substitutes," in *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 2015, pp. 43–52.
[22] C. Q. Nguyen, C. Kreatsoulas, and K. M. Branson, "Meta-Learning GNN Initializations for Low-Resource Molecular Property Prediction," in *ICML Workshop on Graph Representation Learning and Beyond (GRL+)*. ICML, 2020.
[23] I. E. Olatunji, W. Nejdl, and M. Khosla, "Membership Inference Attack on Graph Neural Networks," *CoRR abs/2101.06570*, 2021.
[24] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-Party Deep Network Representation," in *International Joint Conferences on Artifical Intelligence (IJCAI)*. IJCAI, 2016, pp. 1895–1901.
[25] B. Rozemberczki and R. 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)*. ACM, 2020, pp. 1325–1334.
[26] A. Salem, A. Bhattacharya, M. Backes, M. Fritz, and Y. Zhang, "Updates-Leak: Data Set Inference and Reconstruction Attacks in Online Learning," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2020, pp. 1291–1308.
[27] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. 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.
[28] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective Classification in Network Data," *AI Magazine*, 2008.
[29] Y. Shen, X. He, Y. Han, and Y. Zhang, "Model Stealing Attacks Against Inductive Graph Neural Networks," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022, pp. 1175–1192.
[30] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership Inference Attacks Against Machine Learning Models," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, pp. 3–18.
[31] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *International Conference on Learning Representations (ICLR)*, 2018.
[32] B. Wang and N. Z. Gong, "Attacking Graph-based Classification via Manipulating the Graph Structure," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, pp. 2023–2040.
[33] M. Waniek, K. Zhou, Y. Vorobeychik, E. Moro, T. P. Michalak, and T. Rahwan, "Attack Tolerance of Link Prediction Algorithms: How to Hide Your Relations in a Social Network," *CoRR abs/1809.00152*, 2018.
[34] B. Wu, X. Yang, S. Pan, and X. Yuan, "Model Extraction Attacks on Graph Neural Networks: Taxonomy and Realization," *CoRR abs/2010.12751*, 2020.
[35] ——, "Adapting Membership Inference Attacks to GNN for Graph Classification: Approaches and Implications," in *International Conference on Data Mining (ICDM)*. IEEE, 2021.
[36] F. Wu, Y. Long, C. Zhang, and B. Li, "LinkTeller: Recovering Private Edges from Graph Neural Networks via Influence Analysis," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022, pp. 2005–2024.
[37] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, "Adversarial Examples for Graph Data: Deep Insights into Attack and Defense," in *International Joint Conferences on Artifical Intelligence (IJCAI)*. IJCAI, 2019, pp. 4816–4823.
[38] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph Backdoor," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2021.
[39] K. Xu, H. Chen, S. Liu, P. Chen, T. Weng, M. Hong, and X. Lin, "Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective," in *International Joint Conferences on Artifical Intelligence (IJCAI)*. IJCAI, 2019, pp. 3961–3967.
[40] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?" in *International Conference on Learning Representations (ICLR)*, 2019.
[41] H. Zhang, B. Wu, S. Wang, X. Yang, M. Xue, S. Pan, and X. Yuan, "Demystifying Uneven Vulnerability of Link Stealing Attacks against Graph Neural Networks," in *International Conference on Machine Learning (ICML)*. PMLR, 2023.
[42] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong, "Backdoor Attacks to Graph Neural Networks," in *ACM Symposium on Access Control Models and Technologies (SACMAT)*. ACM, 2021, pp. 15–26.
[43] Z. Zhang, M. Chen, M. Backes, Y. Shen, and Y. Zhang, "Inference Attacks Against Graph Neural Networks," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2022, pp. 4543–4560.
[44] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial Attacks on Neural Networks for Graph Data," in *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2018, pp. 2847–2856.
[45] D. Zügner and S. Günnemann, "Adversarial Attacks on Graph Neural Networks via Meta Learning," in *International Conference on Learning Representations (ICLR)*, 2019.

A GNN ARCHITECTURES

In this paper, we focus on four different GNN architectures, i.e., Graph Convolutional Network (GCN), GraphSAGE, Graph Attention Network (GAT), and Graph Isomorphism Network (GIN).

GCN. Kipf et al. [18] propose GCN, which is the most famous and representative GNN method. Combining message computation, aggregation, and update function, the layer of the GCN model can be defined as:

$$h_v^{(l)} = \sigma(\sum_{u \in N(v)} W^{(l)} \frac{h_u^{(l-1)}}{|N(v)|}), \tag{2}$$

where the $W^{(l)}$ is the weight matrix and σ represents the activation function. The GCN layer sums up normalized messages of all neighbors and then applies the activation function to get the representations of node v in the layer l. Note that the self-edges of node v are included in the summation.

GraphSAGE. Hamilton et al. [14] propose GraphSAGE, which can create inductive node embeddings for evolving graphs. Several $\mathcal{A}(\cdot)$ functions have also been proposed. Furthermore, we leverage mean aggregation to take a weighted average of neighbors in this paper. We formulate the GraphSAGE layer as follows:

$$h_v^{(l)} = \sigma(W^{(l)} \cdot CONCAT(h_v^{(l-1)}, \sum_{u \in N(v)} \frac{h_u^{(l-1)}}{|N(v)|})), \tag{3}$$

where *CONCAT* is the concatenate operation.

GAT. Unlike GCN and GraphSAGE, where all neighbors are equally important to a node v, GAT computes attention weights α_{vu} to differentiate the information contribution between neighbors [31]. We formulate the GAT layer as follows:

$$h_v^{(l)} = \sigma(\sum_{u \in N(v)} \alpha_{uv} W^{(l)} h_u^{(l-1)}), \tag{4}$$

where α_{uv} and $W^{(l)}$ are learnable parameters.

GIN. Xu et al. [40] introduce GIN to provide a choice of $\mathcal{A}(\cdot)$ and $\mathcal{U}(\cdot)$ that can make graph message passing neural networks equivalent to the Weisfeiler-Lehman (WL) algorithm. The GIN layer is defined as:

$$h_v^{(l)} = MLP^{(l)}((1 + \epsilon^{(l)}) \cdot h_v^{(l-1)} + \sum_{u \in N(v)} h_u^{(l-1)}), \tag{5}$$

where $\epsilon^{(l)}$ is a learnable parameter and the multi-layer perceptions (*MLP*) can represent the composition of functions.

B GRID SEARCH

Below, we describe the search space of the hyper-parameters of the target, shadow, baseline, and attack models.

Target and shadow models.

- learning rate: $\{0.1, 0.01, 0.001, 0.0001\}$
- number of hidden neurons: $\{64, 128, 256\}$
- dropout rate: $\{0.3, 0.5, 0.7\}$

Table 9: Attack performance of posterior-only attacks on all six datasets when the target model and the shadow model are the same. The average AUC score of five runs is reported.

		M_T and M_S						M_T and M_S			
Dataset	Method	GraphSAGE	GIN	GAT	GCN	Dataset	Method	GraphSAGE	GIN	GAT	GCN
Cora	Attack-0	0.859	0.866	0.882	0.893	Pubmed	Attack-0	0.768	0.752	0.764	0.764
	Attack-1	0.849	0.852	0.884	0.871		Attack-1	0.806	0.800	0.796	0.806
	Attack-2	0.849	0.858	0.878	0.868		Attack-2	0.809	0.805	0.800	0.809
DBLP	Attack-0	0.781	0.748	0.783	0.761	Photo	Attack-0	0.877	0.781	0.838	0.840
	Attack-1	0.821	0.811	0.803	0.815		Attack-1	0.898	0.850	0.901	0.872
	Attack-2	0.867	0.881	0.874	0.837		Attack-2	0.898	0.825	0.907	0.877
CS	Attack-0	0.817	0.795	0.791	0.777	LastFM	Attack-0	0.850	0.829	0.852	0.810
	Attack-1	0.838	0.864	0.851	0.826		Attack-1	0.869	0.868	0.865	0.838
	Attack-2	0.845	0.870	0.853	0.827		Attack-2	0.867	0.881	0.874	0.837

- optimizer: $\{Adam, SGD\}$
- pair-wise operation: {*Hardmard*, *Average*, *Weighted-L1*, *Weighted-L2*, *ALL*}

Baseline and attack models.

- learning rate: $\{0.01, 0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007\}$
- number of hidden layers: $\{2, 3, 4, 5\}$
- number of hidden neurons: $\{4, 8, 16, 32, 64, 128, 256\}$
- batch size: $\{128, 256, 512, 1024\}$
- dropout rate: $\{0.3, 0.5\}$
- optimizer: $\{Adam, SGD\}$
- pair-wise operation: {*Hardmard*, *Average*, *Weighted-L1*, *Weighted-L2*, *ALL*}

ALL denotes the concatenations of all four operations.

C ATTACK SUCCESS ON DIFFERENT TARGET ARCHITECTURES

We conduct evaluations of posterior-only attacks (Attack-0/1/2) on different target architectures, as the choice of GNN target architectures only affects the posterior features in Table 9. The results demonstrate that the proposed attacks remain effective on GIN, GAT, and GCN target models.

D IMPACT OF DIFFERENT ATTACK MODELS

We explore the impact of different attack models in Table 10. Specifically, we consider different layers of attack models (MLP classifiers) for posterior-only attacks. We observe that the effectiveness of posterior-only attacks remains effective.

E IMPACT OF DIFFERENT SHADOW ARCHITECTURES

We report the performance of combined attacks with different shadow model architectures in Table 11, and the results demonstrate that the mismatched shadow model architectures can still enable successful combined attacks.

F ROBUSTNESS ANALYSIS

We compare our attacks and baselines in terms of AUC scores on different groups sorted by four metrics on the Cora (Figure 7), DBLP (Figure 8), Pubmed (Figure 9), Photo (Figure 10), and CS (Figure 11) datasets. We notice that the posteriors can enable robust link stealing attacks. Also, combining the posteriors, inherent graph features and node attributes can result in more robust and better attack performance, especially in groups with lower metric values.

G PEARSON CORRELATION COEFFICIENT

We also present the analysis of robustness using the Pearson correlation coefficient on the Cora dataset (Figure 12). The value of PCC ranges from -1 to 1. Positive numbers represent positive correlations, while negative numbers indicate negative correlations. The higher the absolute values of the PCC, the stronger the correlations between two variables. We take the attack confidence scores to quantify the attack performance. As we can see, for positive pairs, there is a positive correlation between attack performance and these metrics in link stealing attacks, while there is a negative correlation for negative pairs. The results are in line with our intuition that node pairs that have a large overlap in terms of neighbors and high node similarity are more prone to form links, and vice versa. Baseline-0 and Baseline-2 which use node attributes have a high correlation between attack performance and node similarity. Baseline-1 and Baseline-2 both use graph features to design attack input features. The correlations between Baseline-1's attack performance and the three graph features are strong. However, Baseline-2 has almost no correlation between the attack performance and the graph features. We attribute this to the fact that the high-dimensional node attributes have more information to affect the attack performance. Compared to baselines, the PCC values of all posterior-only attacks are closer to 0, which indicates that the posterior-only attacks have little correlation between attack performance, graph features, and node similarity. As node attributes are used to design attack input features in Attack-3, Attack-4, Attack-5, Attack-8, and Attack-9, the correlation between node similarity and attack performance is increased. However, our attacks' correlation with the node similarity metric is still weaker than the baselines that use node attributes. Meanwhile, combined attacks using graph features also yield a boost in the correlation between graph features and attack performance. Similarly, our attacks' correlations with graph features are weaker than the baselines using graph features.

Table 10: Attack performance of posterior-only attacks on six datasets when the target model and the shadow model are GraphSAGE. Different model layers are used in the attack model. The AUC score is reported.

		# Layers						# Layers			
Dataset	Method	2	3	4	5	Dataset	Method	2	3	4	5
Cora	Attack-0	0.858	0.859	0.863	0.861	Pubmed	Attack-0	0.763	0.768	0.766	0.764
	Attack-1	0.842	0.848	0.852	0.847		Attack-1	0.796	0.805	0.808	0.806
	Attack-2	0.843	0.849	0.852	0.847		Attack-2	0.803	0.812	0.814	0.813
DBLP	Attack-0	0.777	0.782	0.776	0.777	Photo	Attack-0	0.880	0.877	0.874	0.868
	Attack-1	0.818	0.821	0.822	0.822		Attack-1	0.899	0.891	0.878	0.887
	Attack-2	0.819	0.822	0.824	0.824		Attack-2	0.898	0.895	0.885	0.870
CS	Attack-0	0.823	0.809	0.810	0.801	LastFM	Attack-0	0.859	0.851	0.839	0.845
	Attack-1	0.840	0.835	0.817	0.813		Attack-1	0.866	0.873	0.865	0.871
	Attack-2	0.847	0.844	0.813	0.807		Attack-2	0.878	0.874	0.864	0.881

Table 11: Attack performance of combined attacks on all six datasets when the target model is GraphSAGE and the shadow model is one of four architectures we mentioned in Section 2. The average AUC score of five runs is reported.

		M_S						M_S			
Dataset	Method	GraphSAGE	GIN	GAT	GCN	Dataset	Method	GraphSAGE	GIN	GAT	GCN
Cora	Attack-3	0.876	0.874	0.880	**0.881**	Pubmed	Attack-3	0.889	**0.896**	0.887	0.891
	Attack-4	0.876	0.865	**0.879**	**0.879**		Attack-4	0.895	0.889	**0.894**	0.888
	Attack-5	0.875	0.868	0.873	**0.879**		Attack-5	**0.897**	0.892	0.892	0.893
	Attack-6	0.882	0.887	0.881	**0.888**		Attack-6	**0.881**	0.878	0.877	0.876
	Attack-7	0.884	0.880	0.881	**0.888**		Attack-7	**0.892**	0.891	0.890	0.889
	Attack-8	**0.882**	**0.882**	**0.882**	0.878		Attack-8	**0.939**	0.938	0.938	0.937
	Attack-9	0.909	0.914	**0.915**	0.914		Attack-9	**0.939**	0.938	0.939	0.938
DBLP	Attack-3	0.834	0.835	**0.839**	0.837	Photo	Attack-3	**0.892**	0.871	0.863	0.875
	Attack-4	0.873	**0.878**	**0.878**	0.873		Attack-4	0.916	0.890	0.896	0.899
	Attack-5	0.872	0.872	**0.878**	0.874		Attack-5	**0.915**	0.900	0.902	0.902
	Attack-6	0.879	0.883	**0.894**	0.879		Attack-6	0.967	0.920	0.956	**0.968**
	Attack-7	**0.903**	0.879	0.876	0.882		Attack-7	**0.968**	0.909	0.952	0.966
	Attack-8	0.924	**0.927**	0.926	0.917		Attack-8	**0.946**	0.925	0.936	0.941
	Attack-9	**0.929**	0.926	0.925	0.921		Attack-9	**0.946**	0.927	0.935	0.943
CS	Attack-3	0.869	**0.872**	0.850	0.843	LastFM	Attack-3	**0.883**	0.876	0.880	0.878
	Attack-4	**0.890**	0.881	0.884	0.874		Attack-4	0.909	0.897	**0.910**	0.890
	Attack-5	**0.893**	0.875	0.885	0.879		Attack-5	**0.911**	0.910	0.909	0.907
	Attack-6	**0.955**	0.913	0.917	0.935		Attack-6	**0.919**	0.891	0.911	0.886
	Attack-7	**0.932**	0.919	0.921	0.929		Attack-7	**0.921**	0.899	0.913	0.874
	Attack-8	**0.945**	0.940	0.943	0.939		Attack-8	**0.929**	0.924	0.923	0.914
	Attack-9	**0.940**	0.936	**0.940**	0.935		Attack-9	**0.930**	0.927	0.927	0.912

The strong correlation can indicate that the attack model overly depends on a specific feature to perform the classification task. Thus, the attack model loses robustness against that feature.

H THE CUMULATIVE DISTRIBUTION FUNCTION OF LEADING PROBABILITY

The cumulative distribution function of leading probability is reported in Figure 13, we observe that the leading probability of the target model's outputs dominates the rest, especially in Attack-1 and Attack-2.

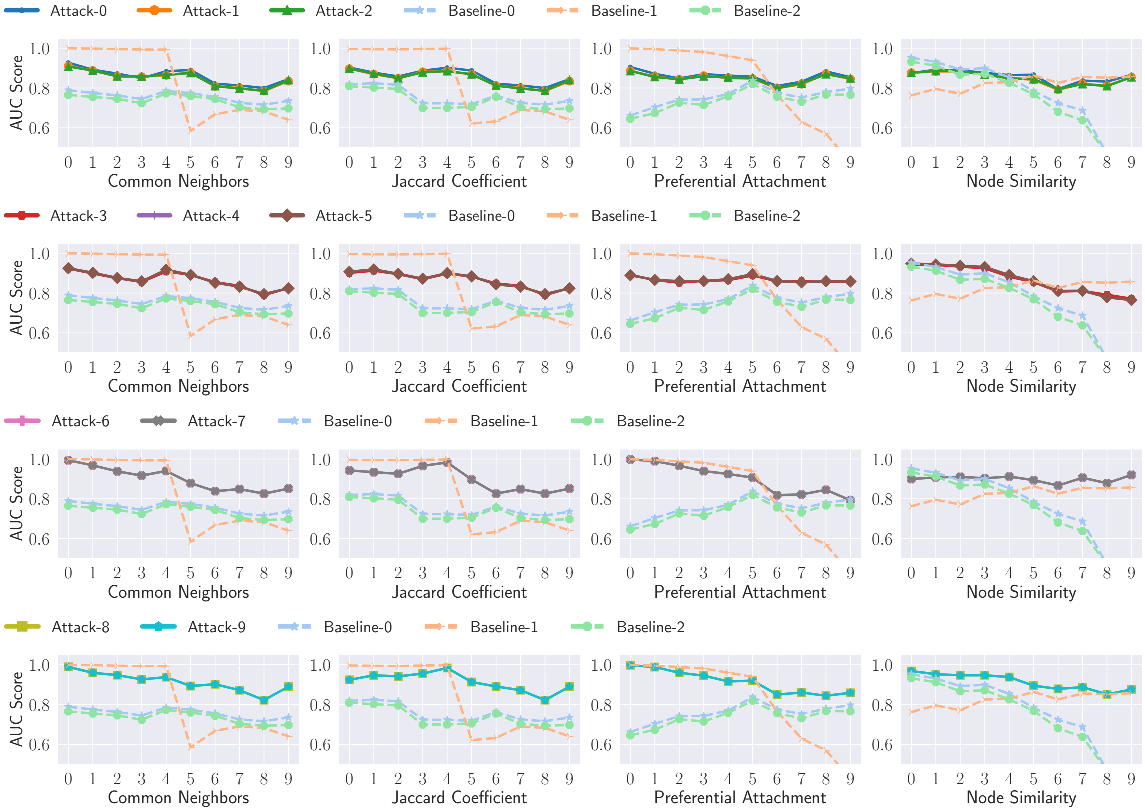


Figure 7: AUC score for our attacks and baselines on ten different groups on the Cora dataset. The ten groups are formed by categorizing all existing links in G_{Target}^{Train} (the positive pairs in D_{Attack}^{Test}) based on node attributes and three graph metrics, respectively. The x-axis represents different groups in descending order of their corresponding metric values. The y-axis represents the AUC scores. Each column represents one metric. The first to fourth rows are $A^*_{(p,\cdot,\cdot)}$, $A^*_{(p,n,\cdot)}$, $A^*_{(p,\cdot,g)}$, and $A^*_{(p,n,g)}$, respectively.

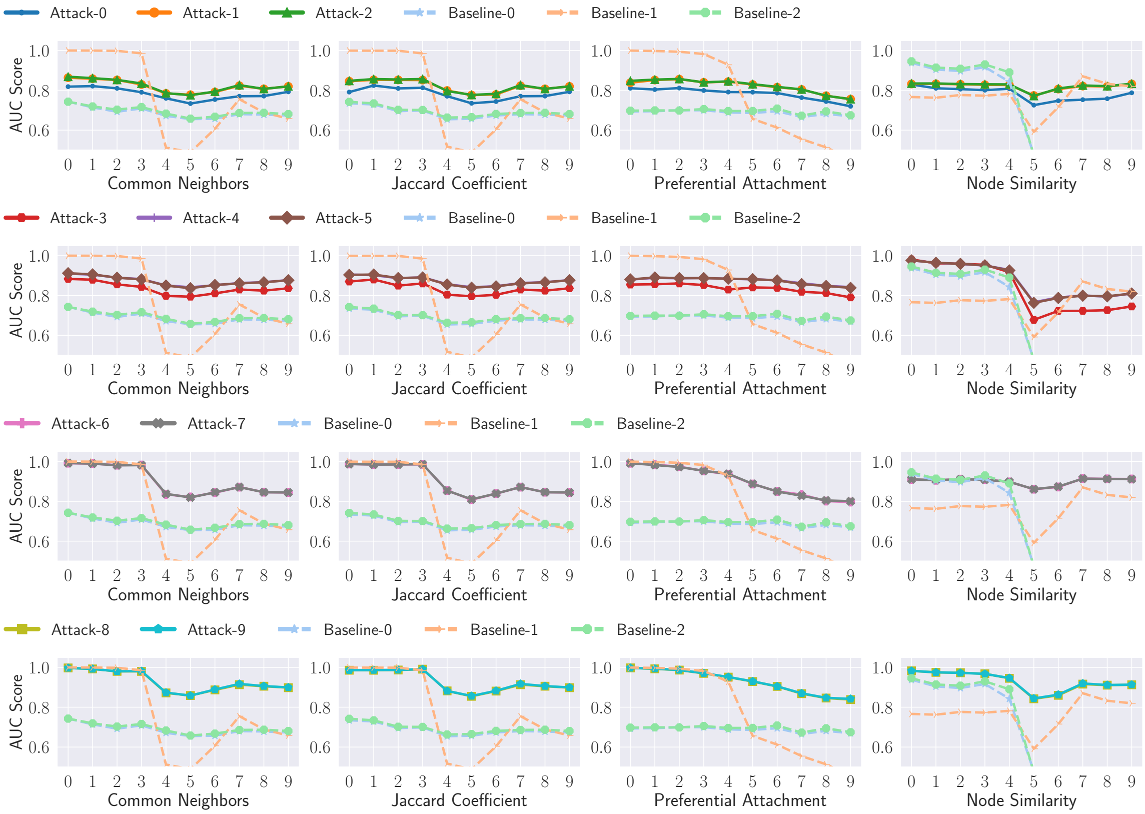


Figure 8: AUC score for our attacks and baselines on ten different groups on the DBLP dataset. The ten groups are formed by categorizing all existing links in G^{Train}_{Target} (the positive pairs in D^{Test}_{Attack}) based on node attributes and three graph metrics, respectively. The x-axis represents different groups in descending order of their corresponding metric values. The y-axis represents the AUC scores. Each column represents one metric. The first to fourth rows are $A^*_{(p,\ \cdot,\ \cdot)}$, $A^*_{(p,n,\ \cdot)}$, $A^*_{(p,\ \cdot,g)}$, and $A^*_{(p,n,g)}$, respectively.

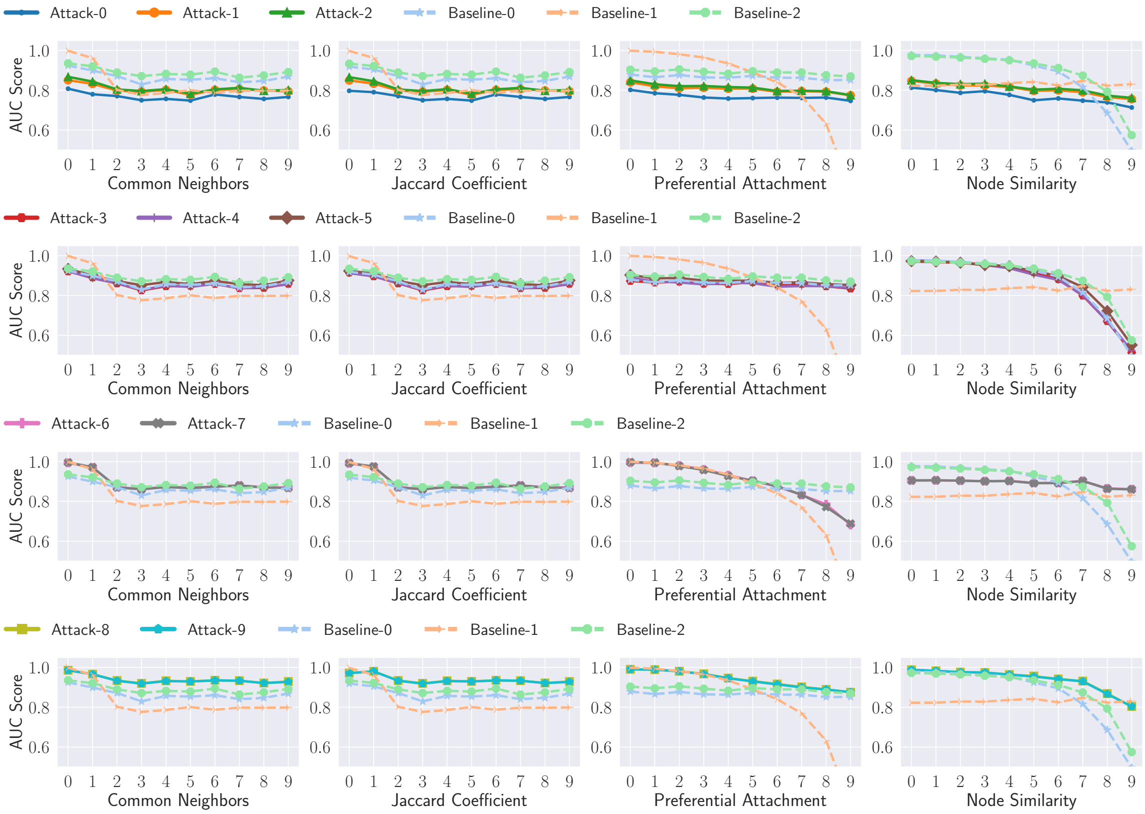


Figure 9: AUC score for our attacks and baselines on ten different groups on the Pubmed dataset. The ten groups are formed by categorizing all existing links in G^{Train}_{Target} (the positive pairs in D^{Test}_{Attack}) based on node attributes and three graph metrics, respectively. The x-axis represents different groups in descending order of their corresponding metric values. The y-axis represents the AUC scores. Each column represents one metric. The first to fourth rows are $A^*_{(p,\ \cdot,\ \cdot)}$, $A^*_{(p,n,\ \cdot)}$, $A^*_{(p,\ \cdot,g)}$, and $A^*_{(p,n,g)}$, respectively.

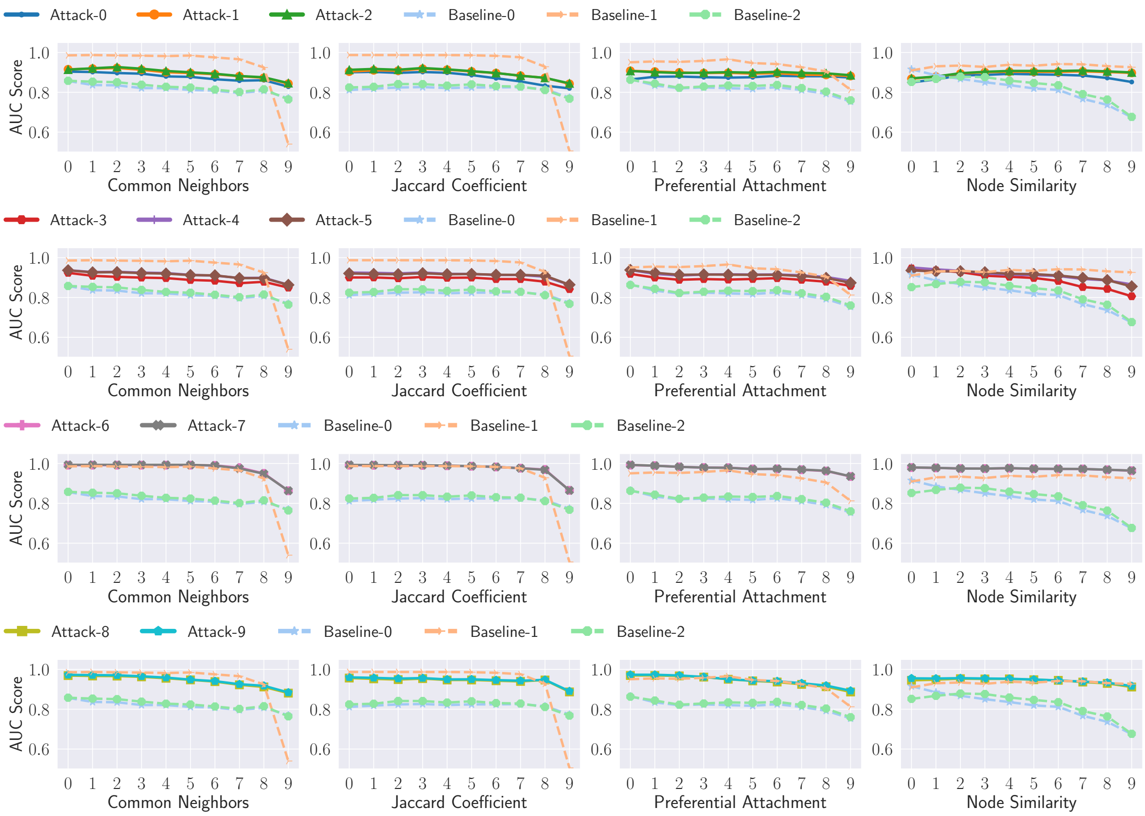


Figure 10: AUC score for our attacks and baselines on ten different groups on the Photo dataset. The ten groups are formed by categorizing all existing links in G^{Train}_{Target} (the positive pairs in D^{Test}_{Attack}) based on node attributes and three graph metrics, respectively. The x-axis represents different groups in descending order of their corresponding metric values. The y-axis represents the AUC scores. Each column represents one metric. The first to fourth rows are $A^*_{(p,\cdot,\cdot)}$, $A^*_{(p,n,\cdot)}$, $A^*_{(p,\cdot,g)}$, and $A^*_{(p,n,g)}$, respectively.

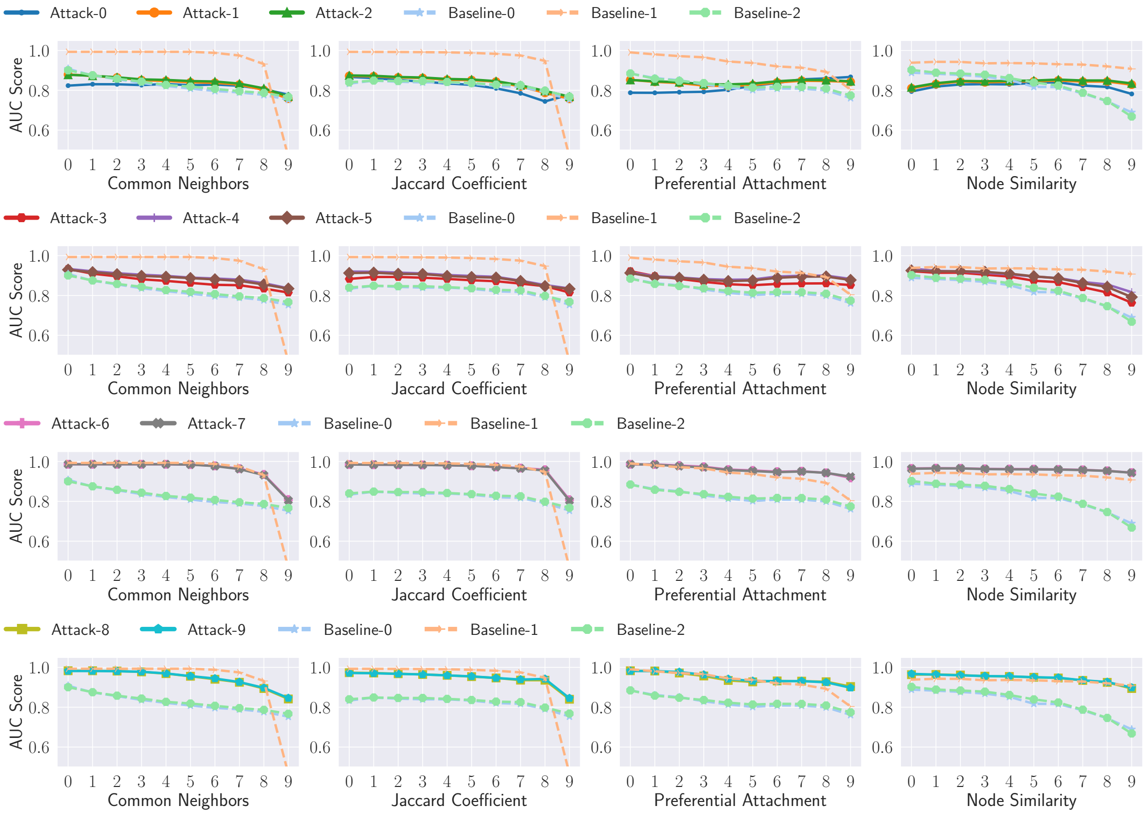
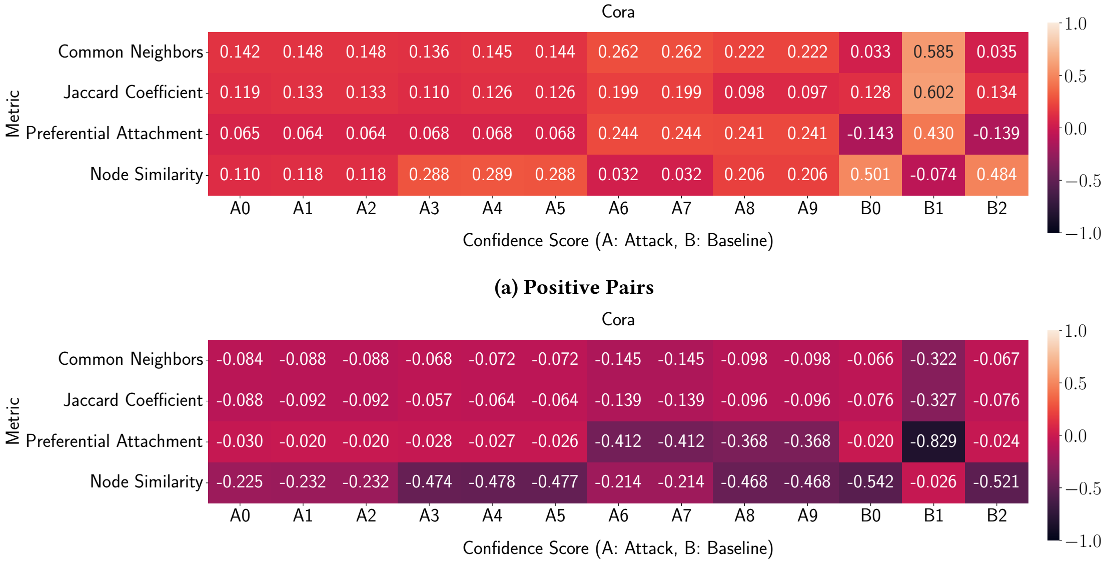


Figure 11: AUC score for our attacks and baselines on ten different groups on the CS dataset. The ten groups are formed by categorizing all existing links in G^{Train}_{Target} (the positive pairs in D^{Test}_{Attack}) based on node attributes and three graph metrics, respectively. The x-axis represents different groups in descending order of their corresponding metric values. The y-axis represents the AUC scores. Each column represents one metric. The first to fourth rows are $A^*_{(p,\ \cdot,\ \cdot)}$, $A^*_{(p,n,\ \cdot)}$, $A^*_{(p,\ \cdot,g)}$, and $A^*_{(p,n,g)}$, respectively.



(a) Positive Pairs

(b) Negative Pairs

Figure 12: Pearson correlation coefficient (PCC) between four metrics and attack performance. We use PCC to measure the correlation between the attack performance and metric values. The dataset is fixed to Cora. The x-axis denotes ten link stealing attacks and three baselines. The y-axis represents metrics of interest. The value of PCC ranges from -1 to 1. Positive numbers represent positive correlations, while negative numbers indicate negative correlations. The higher the absolute values of the PCC, the stronger are the correlations between the two variables. We take the attack confidence scores to quantify the attack performance.

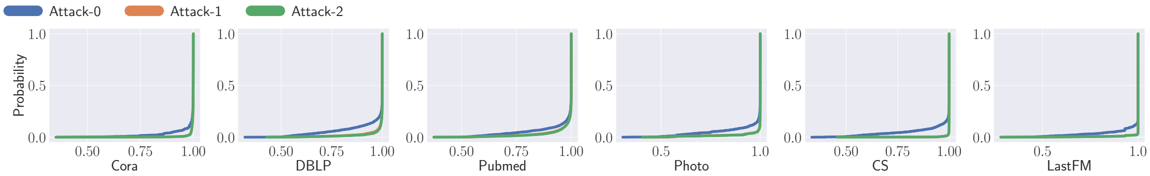


Figure 13: The cumulative distribution function of the leading probability of the target model's outputs.