Stealing Links from Graph Neural Networks

Xinlei He[1] Jinyuan Jia[2] Michael Backes[1] Neil Zhenqiang Gong[2] Yang Zhang[1]

[1]CISPA Helmholtz Center for Information Security [2]Duke University

Abstract

Graph data, such as chemical networks and social networks, may be deemed confidential/private because the data owner often spends lots of resources collecting the data or the data contains sensitive information, e.g., social relationships. Recently, neural networks were extended to graph data, which are known as *graph neural networks (GNNs)*. Due to their superior performance, GNNs have many applications, such as healthcare analytics, recommender systems, and fraud detection. In this work, we propose the first attacks to steal a graph from the outputs of a GNN model that is trained on the graph. Specifically, given a black-box access to a GNN model, our attacks can infer whether there exists a link between any pair of nodes in the graph used to train the model. We call our attacks *link stealing attacks*. We propose a threat model to systematically characterize an adversary's background knowledge along three dimensions which in total leads to a comprehensive taxonomy of 8 different link stealing attacks. We propose multiple novel methods to realize these 8 attacks. Extensive experiments on 8 real-world datasets show that our attacks are effective at stealing links, e.g., AUC (area under the ROC curve) is above 0.95 in multiple cases. Our results indicate that the outputs of a GNN model reveal rich information about the structure of the graph used to train the model.

1 Introduction

Graph is a powerful tool to model the complex relationships between entities. For instance, in healthcare analytics, protein-protein interactions can be modeled as a graph (called a *chemical network*); and a social network can be modeled as a graph, where nodes are users and edges indicate certain social relationships among them. A graph may be treated as a data owner's intellectual property because the data owner may spend a lot of resources collecting the graph, e.g., collecting a chemical network often involves expensive and resource-consuming chemical experiments. Moreover, a graph may also contain sensitive user information, e.g., private social relationships among users.

Recently, a family of machine learning techniques known as *graph neural networks (GNNs)* was proposed to analyze graphs. We consider GNNs for *node classification*. Specifically, given a graph, attributes of each node in the graph, and a small number of node labels, a GNN model is trained and can predict the label of each remaining unlabeled node. Due to their superior performance, we have seen growing applications of GNNs in various domains, such as healthcare analytics [18, 22], recommender systems [19], and fraud detection [65]. However, the security and privacy implications of training GNNs on graphs are largely unexplored.

Our Contributions. In this work, we take the first step to study the security and privacy implications of training GNNs on graphs. In particular, we propose the first attacks to steal a graph from the outputs of a GNN model trained on the graph. We call our attacks *link stealing attacks*. Specifically, given a black-box access to a target GNN model, our attacks aim to predict whether there exists a link between any pair of nodes in the graph used to train the target GNN model. Our attacks reveal serious concerns on the intellectual property, confidentiality, and/or privacy of graphs when training GNNs on them. For instance, our attacks violate the intellectual property of the data owner when it spends lots of resources collecting the graph; and our attacks violate user privacy when the graph contains sensitive social relationships among users [2, 23].

Adversary's Background Knowledge: We refer to the graph and nodes' attributes used to train the target GNN model as the *target dataset*. We characterize an adversary's background knowledge along three dimensions, including the target dataset's *nodes' attributes*, the target dataset's *partial graph*, and an auxiliary dataset (called *shadow dataset*) which also contains its own graph and nodes' attributes. An adversary may or may not have access to each of the three dimensions. Therefore, we obtain a comprehensive taxonomy of a threat model, in which adversaries can have 8 different types of background knowledge.

Attack Methodology: We design an attack for each of the 8 different types of background knowledge, i.e., we propose 8 link stealing attacks in total. The key intuition of our attacks is that two nodes are more likely to be linked if they share more similar attributes and/or predictions from the target GNN model. For instance, when the adversary only has the target dataset's nodes' attributes, we design an unsupervised attack by calculating the distance between two nodes' attributes. When the target dataset's partial graph is available, we use supervised learning to train a binary classifier as our attack model with features summarized from two nodes' attributes and predictions obtained from the black-box access to the target GNN model. When the adversary has a shadow dataset, we propose a *transferring attack* which transfers the knowledge from the shadow dataset to the target dataset to mount our attack.

Evaluation: We evaluate our 8 attacks using 8 real-world datasets. First, extensive experiments show that our attacks can effectively steal links. In particular, our attacks achieve high AUCs (area under the ROC curve). This demonstrates that the predictions of a target GNN model encode rich information about the structure of a graph that is used to train the model, and our attacks can exploit them to steal the graph structure. Second, we observe that more background knowledge leads to better attack performance in general. For instance, on the Citeseer dataset [35], when an adversary has all the three dimensions of the background knowledge, our attack achieves 0.977 AUC. On the same dataset, when the adversary only has nodes' attributes, the AUC is 0.878. Third, we find that the three dimensions of background knowledge have different impacts on our attacks. Specifically, the target dataset's partial graph has the strongest impact followed by nodes' attributes, the shadow dataset, on the other hand, has the weakest impact. Fourth, our transferring attack can achieve high AUCs. Specifically, our transferring attack achieves better performance if the shadow dataset comes from the same domain as the target dataset, e.g., both of them are chemical networks. We believe this is due to the fact that graphs from the same domain have similar structures, which leads to less information loss during transferring. Fifth, our attacks outperform conventional link prediction methods [24, 40], which aim to predict links between nodes based on a partial graph.

In summary, we make the following contributions.

- We propose the first link stealing attacks against graph neural networks.
- We propose a threat model to comprehensively characterize an adversary's background knowledge along three dimensions. Moreover, we propose 8 link stealing attacks for adversaries with different background knowledge.
- We extensively evaluate our 8 attacks on 8 real-world datasets. Our results show that our attacks can steal links from a GNN model effectively.

2 Graph Neural Networks

Many important real-world datasets come in the form of graphs or networks, e.g., social networks, knowledge graph, and chemical networks. Therefore, it is urgent to develop machine learning algorithms to fully utilize graph data. To this end, a new family of machine learning algorithms, i.e., graph neural networks (GNNs), has been proposed and shown superior performance in various tasks [1, 14, 35, 62].

Training a GNN Model. Given a graph, attributes for each node in the graph, and a small number of labeled nodes, GNN trains a neural network to predict labels of the remaining unlabeled nodes via analyzing the graph structure and node attributes. Formally, we define the *target dataset* as $\mathcal{D} = (\mathcal{A}, \mathcal{F})$, where $\mathcal{A}$ is the adjacency matrix of the graph and $\mathcal{F}$ contains all nodes' attributes. Specifically, $\mathcal{A}_{uv}$ is an element in $\mathcal{A}$: If there exists an edge between node u and node v, then $\mathcal{A}_{uv} = 1$, otherwise $\mathcal{A}_{uv} = 0$. Moreover, $\mathcal{F}_u$ represents the attributes of u. $\mathcal{V}$ is a set containing all nodes in the graph. Note that we consider undirected graphs in this paper, i.e., $\forall u, v \in \mathcal{V}, \mathcal{A}_{uv} = \mathcal{A}_{vu}$.

A GNN method iteratively updates a node's features via aggregating its neighbors' features using a neural network, whose last layer predicts labels for nodes. Different GNN methods use slightly different aggregation rules. For instance, *graph convolutional network (GCN)*, the most representative and well-established GNN method [35], uses a multi-layer neural network whose architecture is determined by the graph structure. Specifically, each layer obeys the following propagation rule to aggregate the neighboring features:

$$H^{(k+1)} = \sigma(\tilde{Q}^{-\frac{1}{2}} \tilde{\mathcal{A}} \tilde{Q}^{-\frac{1}{2}} H^{(k)} W^{(k)}), \quad (1)$$

where $\tilde{\mathcal{A}} = \mathcal{A} + I$ is the adjacency matrix of the graph with self-connection added, i.e., I is the identity matrix. $\tilde{Q}^{-\frac{1}{2}} \tilde{\mathcal{A}} \tilde{Q}^{-\frac{1}{2}}$ is the symmetric normalized adjacency matrix and $\tilde{Q}_{uu} = \sum_u \tilde{\mathcal{A}}_{uv}$. Moreover, $W^{(k)}$ is the trainable weight matrix of the kth layer and $\sigma(\cdot)$ is the activation function to introduce non-linearity, such as ReLU. As the input layer, we have $H^{(0)} = \mathcal{F}$. When the GCN uses a two-layer neural network, the GCN model can be described as follows:

$$softmax(\tilde{Q}^{-\frac{1}{2}} \tilde{\mathcal{A}} \tilde{Q}^{-\frac{1}{2}} \sigma(\tilde{Q}^{-\frac{1}{2}} \tilde{\mathcal{A}} \tilde{Q}^{-\frac{1}{2}} \mathcal{F} W^{(0)}) W^{(1)}). \quad (2)$$

Note that in most of the paper, we focus on two-layer GCN. Later, we show that our attack can be also performed on other types of GNNs, including GraphSAGE [27] and GAT [62] (see Section 5).

Prediction in a GNN Model. Since all nodes' attributes and the whole graph have been fed into the GNN model in the training phase to predict the label of a node, we only need to provide the node's ID to the trained model and obtain the prediction result. We assume the prediction result is a posterior distribution (called *posteriors*) over the possible labels

Table 1: List of notations.

Notation	Description
$\mathcal{D}$	Target dataset
$\mathcal{A}$	Graph of $\mathcal{D}$ represented as adjacency matrix
$\mathcal{A}^*$	Partial graph of $\mathcal{D}$
$\mathcal{F}$	Nodes' attributes of $\mathcal{D}$
$\mathcal{V}$	Set of nodes of $\mathcal{D}$
f	Target model
g	Reference model
$f(u)$	u's posteriors from the target model
$g(u)$	u's posteriors from the reference model
$\mathcal{D}'$	Shadow dataset
f'	Shadow target model
g'	Shadow reference model
$\mathcal{K}$	Adversary's knowledge
$d(\cdot,\cdot)$	Distance metric
$\Psi(\cdot,\cdot)$	Pairwise vector operations
$e(f(u))$	Entropy of $f(u)$

for the node. Our work shows that such posteriors reveal rich information about the graph structure: As mentioned before, a GNN essentially learns a node's features via aggregating its neighbors' features, if two nodes are connected, then their posteriors should be similar. We leverage this to build our attack models. We further use f to denote the target GNN model and $f(u)$ to represent the posteriors of node u. For presentation purposes, we summarize the notations introduced here and in the following sections in Table 1.

3 Problem Formulation

In this section, we first propose a threat model to characterize an adversary's background knowledge. Then, we formally define our link stealing attack.

3.1 Threat Model

Adversary's Goal. An adversary's goal is to infer whether a given pair of nodes u and v are connected in the target dataset. Inferring links between nodes leads to a severe privacy threat when the links represent sensitive relationship between users in the context of social networks. Moreover, links may be confidential and viewed as a model owner's intellectual property because the model owner may spend lots of resources collecting the links, e.g., it requires expensive medical/chemical experiments to determine the interaction/link between two molecules in a chemical network. Therefore, inferring links may also compromise a model owner's intellectual property.

Adversary's Background Knowledge. First, we assume an adversary has a black-box access to the target GNN model. In other words, the adversary can only obtain nodes' posteriors by querying the target model f. This is the most difficult setting for the adversary [52, 54, 56]. An adversary can have a black-box access to a GNN model when an organization uses GNN tools from another organization (viewed as an adversary) or the GNN model prediction results are shared among different departments within the same organization. For instance, suppose a social network service provider leverages another company's tool to train a GNN model for fake-account detection, the provider often needs to send the prediction results of (some) nodes to the company for debugging or refining purposes. In such a scenario, the security company essentially has a black-box access to the GNN model. Note that the graph structure is already revealed to the adversary if she has a white-box access to the target GNN model as the GNN model architecture is often based on the graph structure.

Then, we characterize an adversary's background knowledge along three dimensions:

- **Target Dataset's Nodes' Attributes, denoted by $\mathcal{F}$.** This background knowledge characterizes whether the adversary knows nodes' attributes $\mathcal{F}$ in $\mathcal{D}$. We also assume that the adversary knows labels of a small subset of nodes.

- **Target Dataset's Partial Graph, denoted by $\mathcal{A}^*$.** This dimension characterizes whether the adversary knows a subset of links in the target dataset $\mathcal{D}$. Since the goal of link stealing attack is to infer whether there exists an edge/link between a pair of nodes, the partial graph can be used as ground truth edges to train the adversary's attack model.

- **A Shadow Dataset, denoted by $\mathcal{D}'$.** This is a dataset which contains its own nodes' attributes and graph. The adversary can use this to build a GNN model, referred to as *shadow target model* (denoted by f') in order to perform a transferring attack. It is worth noting that the shadow dataset does not need to come from the same domain of the target dataset. For instance, the shadow dataset can be a chemical network, while the target dataset can be a citation network. However, results in Section 5 show that same-domain shadow dataset indeed leads to better transferring attack performance.

We denote the adversary's background knowledge as a triplet:

$$\mathcal{K} = (\mathcal{F}, \mathcal{A}^*, \mathcal{D}').$$

Whether the adversary has each of the three items is a binary choice, i.e., yes or no. Therefore, we have a comprehensive taxonomy with 8 different types of background knowledge, which leads to 8 different link stealing attacks. Table 2 summarizes our attack taxonomy.

3.2 Link Stealing Attack

After describing our threat model, we can formally define our link stealing attack as follows:

Table 2: Attack taxonomy. ✓ (×) means the adversary has (does not have) the knowledge.

Attack	$\mathcal{F}$	$\mathcal{A}^*$	$\mathcal{D}'$	Attack	$\mathcal{F}$	$\mathcal{A}^*$	$\mathcal{D}'$
Attack-0	×	×	×	Attack-4	×	✓	✓
Attack-1	×	×	✓	Attack-5	✓	×	✓
Attack-2	✓	×	×	Attack-6	✓	✓	×
Attack-3	×	✓	×	Attack-7	✓	✓	✓

Definition 1 (Link Stealing Attack)**.** *Given a black-box access to a GNN model that is trained on a target dataset, a pair of nodes u and v in the target dataset, and an adversary's background knowledge $\mathcal{K}$, link stealing attack aims to infer whether there is a link between u and v in the target dataset.*

4 Attack Taxonomy

In this section, we present the detailed constructions of all the 8 attacks in Table 2. Given different knowledge $\mathcal{K}$, the adversary can conduct their attacks in different ways. However, there are two problems that exist across different attacks.

The first problem is *node pair order*. As we consider undirected graph, when the adversary wants to predict whether there is a link between two given nodes u and v, the output should be the same regardless of the input node pair order.

The second problem is *dimension mismatch*. The shadow dataset and the target dataset normally have different dimensions with respect to attributes and posteriors (as they are collected for different classification tasks). For transferring attacks that require the adversary to transfer information from the shadow dataset to the target dataset, it is crucial to keep the attack model's input features' dimension consistent no matter which shadow dataset she has.

We will discuss how to solve these two problems during the description of different attacks. For presentation purposes, features used in our supervised attacks and transferring attacks are summarised in Table 3.

4.1 Attack Methodologies

Attack-0: $\mathcal{K} = (\times, \times, \times)$. We start with the most difficult setting for the adversary, that is she has no knowledge of the target dataset's nodes' attributes, partial graph, and a shadow dataset. All she has is the posteriors of nodes obtained from the target model f (see Section 2).

As introduced in Section 2, GNN essentially aggregates information for each node from its neighbors. This means if there is a link between two nodes, then their posteriors obtained from the target model should be closer. Following this intuition, we propose an unsupervised attack. More specifically, to predict whether there is a link between u and v , we calculate the distance between their posteriors, i.e., $d(f(u), f(v))$, as the predictor.

We have in total experimented with 8 common distance metrics: Cosine distance, Euclidean distance, Correlation distance, Chebyshev distance, Braycurtis distance, Canberra distance, Manhattan distance, and Square-euclidean distance. Their formal definitions are in Table 13 in Appendix. It is worth noting that all distance metrics we adopt are symmetric, i.e., $d(f(u), f(v)) = d(f(v), f(u))$, this naturally solves the problem of *node pair order*.

Since the attack is unsupervised, to make a concrete prediction, the adversary needs to manually select a threshold depending on application scenarios. To evaluate our attack, we mainly use AUC which considers a set of thresholds as previous works [2, 21, 26, 32, 54, 70]. In addition, we propose a threshold estimation method based on clustering (see Section 5 for more details).

Attack-1: $\mathcal{K} = (\times, \times, \mathcal{D}')$. In this attack, we broaden the adversary's knowledge with a shadow dataset, i.e., $\mathcal{D}'$. This means the adversary can train a classifier for a supervised attack, more specifically, a *transferring attack*. She first constructs a shadow target model f' with $\mathcal{D}'$. Then, she derives the training data from f' to train her attack model.

The adversary cannot directly use the posteriors obtained from the shadow target model as features to train her attack model, as the shadow dataset and the target dataset very likely have different numbers of labels, i.e., the corresponding posteriors are in different dimensions. This is the dimension mismatch problem mentioned before. To tackle this, we need to design features over posteriors.

As discussed in Attack-0, for any dataset, if two nodes are linked, then their posteriors obtained from the target model should be similar. This means if the attack model can capture the similarity of two nodes' posteriors from the shadow target model, it can transfer the information to the target model.

We take two approaches together to design features. The first approach is measuring distances between two nodes' posteriors. To this end, for each pair of nodes u' and v' from the shadow dataset $\mathcal{D}'$, we adopt the same set of 8 metrics used in Attack-0 (formal definitions are listed in Table 13) to measure their posteriors $f'(u')$ and $f'(v')$'s distances, and concatenate these different distances together. This leads to an 8-dimension vector.

The second approach is to use entropy to describe each posterior inspired by previous works [32,42]. Formally, for the posterior of node u' obtained from the shadow target model f', its entropy is defined as the following.

$$e(f'(u')) = -\sum_i f'_i(u') log(f'_i(u')) \tag{3}$$

where $f'_i(u')$ denotes the i-th element of $f'(u')$. Then, for each pair of nodes u' and v' from the shadow dataset, we obtain two entropies $e(f'(u'))$ and $e(f'(v'))$. To eliminate the node pair order problems for these entropies, we further take the approach of Grover and Leskovec [25], by applying pairwise vector operation, denoted by $\Psi(\cdot, \cdot)$. In total, we have used

Table 3: Features adopted by our supervised attacks (Attack-3 and Attack 6) and transferring attacks (Attack-1, Attack-4, Attack-5, and Attack-7). Here, $(*)$ means the features are extracted from the shadow dataset in the training phase, and $(\star)$ means the features are extracted from both the shadow dataset and the target dataset (its partial graph) in the training phase. $d(\cdot,\cdot)$ represents distance metrics defined in Table 13, $\Psi(\cdot,\cdot)$ represents the pairwise vector operations defined in Table 14. Note that the features used in these attack models include all the distance metrics and pairwise vector operations.

Attack	$d(f(u),f(v))$	$\Psi(f(u),f(v))$	$\Psi(e(f(u)),e(f(v)))$	$d(g(u),g(v))$	$\Psi(g(u),g(v))$	$\Psi(e(g(u)),e(g(v)))$	$d(\mathcal{F}_u,\mathcal{F}_v)$	$\Psi(\mathcal{F}_u,\mathcal{F}_v)$
Attack-1 $*$	✓	×	✓	×	×	×	×	×
Attack-3	✓	✓	✓	×	×	×	×	×
Attack-4 $\star$	✓	×	✓	×	×	×	×	×
Attack-5 $*$	✓	×	✓	✓	×	✓	✓	×
Attack-6	✓	✓	✓	✓	✓	✓	✓	✓
Attack-7 $\star$	✓	×	✓	✓	×	✓	✓	×

all the 4 operations defined in Table 14 (in Appendix) for our attack. Note that these operations in Table 14 are applied on two single numbers, i.e., scalars, in this attack. However, they can also be applied to vectors and we will adopt them again on posteriors and nodes' attributes in other attacks.

In total, the features used for training the attack model is assembled with 8 different distances between two nodes' posteriors from the shadow target model and 4 features obtained from pairwise vector operations between two nodes' posteriors' entropies. Regarding labels for the training set, the adversary uses all the links in $\mathcal{D}'$ and samples the same number of node pairs that are not linked (see Section 5 for more details). We adopt an MLP as our attack model.

Attack-2: $\mathcal{K} = (\mathcal{F}, \times, \times)$**.** In this attack, we assume that the adversary has the knowledge of the target dataset's nodes' attributes $\mathcal{F}$. Since the adversary has no knowledge of the partial graph and a shadow dataset, her attack here is also unsupervised (similar to Attack-0). We again rely on the distance metrics to perform our attack. For each pair of nodes u and v from the target dataset, we consider four types of information to measure distance with all the metrics listed in Table 13. Similar to Attack-0, we experimentally decide which is the most suitable distance metric for Attack-2.

- $d(f(u), f(v))$. The first type is the same as the method for Attack-0, i.e., distance between posteriors of u and v from the target model f, i.e., $f(u)$ and $f(v)$.

- $d(\mathcal{F}_u, \mathcal{F}_v)$. The second type is calculating the pairwise distance over u and v's attributes $\mathcal{F}_u$ and $\mathcal{F}_v$.

- $d(f(u), f(v)) - d(g(u), g(v))$. For the third type, since we have the target model's nodes' attributes (as well as a subset of their corresponding labels), we train a separate MLP model, namely *reference model* (denoted by g). Our intuition is that if two nodes are connected, the distance between their posteriors from the target model should be smaller than the corresponding distance from the reference model. Therefore, we calculate $d(f(u), f(v)) - d(g(u), g(v))$ to make prediction.

- $d(g(u), g(v))$. For the fourth type, we measure the distance over u and v's posteriors from the reference model.

Attack-3: $\mathcal{K} = (\times, \mathcal{A}^*, \times)$**.** In this scenario, the adversary has access to the partial graph $\mathcal{A}^*$ of the target dataset. For the attack model, we rely on links from the known partial graph as the ground truth label to train an attack model (we again adopt an MLP). Features used for Attack-3 are summarized in Table 3. For each pair of nodes u and v from the target dataset, we calculate the same set of features proposed for Attack-1 on their posteriors and posteriors' entropies. Besides, since we can directly train the attack model on the partial target graph (i.e., we do not face the dimension mismatch problem), we further define new features by adopting the pairwise vector operations listed in Table 14 to $f(u)$ and $f(v)$.

Attack-4: $\mathcal{K} = (\times, \mathcal{A}^*, \mathcal{D}')$**.** In this attack, the adversary has the knowledge of the partial graph $\mathcal{A}^*$ of the target dataset and a shadow dataset $\mathcal{D}'$. To take both knowledge into consideration, for each pair of nodes either from the shadow dataset or the partial graph of the target dataset, we calculate the same set of features over posteriors as proposed in Attack-1. This means the only difference between Attack-4 and Attack-1 is that the training set for Attack-4 also includes information from the target dataset's partial graph (see Table 3).

Different from Attack-3, Attack-4 cannot perform the pairwise vector operations to $f(u)$ and $f(v)$. This is due to the dimension mismatch problem as the adversary needs to take both $\mathcal{A}^*$ and $\mathcal{D}'$ into account for her attack.

Attack-5: $\mathcal{K} = (\mathcal{F}, \times, \mathcal{D}')$**.** In this attack, the adversary has the knowledge of the target model's nodes' attributes $\mathcal{F}$ and a shadow dataset $\mathcal{D}'$. As we do not have $\mathcal{A}^*$ to train the attack model, we need to rely on the graph of the shadow dataset. To this end, we first calculate the same set of features used for Attack-1. Moreover, as we have the target dataset's nodes' attributes, we further build a reference model (as in Attack-2), and also a shadow reference model in order to transfer more knowledge from the shadow dataset for the attack. For this, we build the same set of features as in Attack-1 over the posteriors obtained from the shadow reference model, i.e., the distance of posteriors (Table 13) and pairwise vector operations performed on posteriors' entropies (Table 14). In addition, we also calculate the 8 different distances over the shadow dataset's nodes' attributes.

Attack-6: $\mathcal{K} = (\mathcal{F}, \mathcal{A}^*, \times)$**.** In this scenario, the adversary has the access to the target dataset's nodes' attributes $\mathcal{F}$ and the partial target graph $\mathcal{A}^*$. As a supervised learning setting, we build an MLP considering links from the partial graph as the ground truth label. The adversary first adopts the same set of features defined over posteriors obtained from the target model as proposed in Attack-3. Then, the adversary builds a reference model over the target dataset's nodes' attributes, and calculate the same set of features over posteriors obtained from the reference model. In the end, we further calculate the distances of the target dataset's nodes' attributes as another set of features.

Attack-7: $\mathcal{K} = (\mathcal{F}, \mathcal{A}^*, \mathcal{D}')$**.** This is the last attack with the adversary having all three knowledge. The set of features for this attack is the same as the ones used in Attack-5 (Table 3). The only difference lies in the training phase, we can use the partial graph from the target dataset together with the graph from the shadow dataset as the ground truth. We expect this leads to better performance than the one for Attack-5. However, this attack also relies on the information of the shadow dataset, thus, the features used here are a subset of the ones for Attack-6, this is similar to the difference between Attack-4 and Attack-3. Note that if the adversary does not take the shadow dataset into consideration, this scenario is equivalent to the one for Attack-6.

4.2 Summary

We propose 8 attack scenarios with the combination of the knowledge that the adversary could have. They could be divided into three categories.

The first category is unsupervised attacks, i.e., Attack-0 and Attack-2, where the adversary does not have the knowledge about the partial graph from the target dataset or a shadow dataset. In these scenarios, the adversary can use distance metrics for posteriors or nodes' attributes to infer the link.

The second category is the supervised attacks, including Attack-3 and Attack-6, where the adversary has the knowledge of the partial graph from the target dataset but does not have a shadow dataset. In these scenarios, the adversary can use different distances and pairwise vector operations over nodes' posteriors (and the corresponding entropies) from the target model and their attributes to build features.

The third category is the transferring attacks (supervised), including Attack-1, Attack-4, Attack-5, and Attack-7, where the adversary has the knowledge of a shadow dataset. In these scenarios, the adversary can use distance metrics over posteriors/nodes' attributes and pairwise operations over posteriors' entropies as the bridge to transfer the knowledge from the shadow dataset to perform link stealing attacks. It is worth noting that for Attack-4 and Attack-7, if the adversary leaves the shadow dataset out of consideration, they will not have the dimension mismatch problem and can take the same attack methods as Attack-3 and Attack-6, respectively.

5 Evaluation

This section presents the evaluation results of our 8 attacks. We first introduce our experimental setup. Then, we present detailed results for different attacks. Finally, we summarize our experimental findings.

5.1 Experimental Setup

Datasets. We utilize 8 public datasets, including Citeseer [35], Cora [35], Pubmed [35], AIDS [51], COX2 [59], DHFR [59], ENZYMES [15], and PROTEINS_full [5], to conduct our experiments. These datasets are widely used as benchmark datasets for evaluating GNNs [17, 18, 35, 62]. Among them, Citeseer, Cora, and Pubmed are citation datasets with nodes representing publications and links indicating citations among these publications. The other five datasets are chemical datasets, each node is a molecule and each link represents the interaction between two molecules. All these datasets have nodes' attributes and labels.

Datasets Configuration. For each dataset, we train a target model and a reference model. In particular, we randomly sample 10% nodes and use their ground truth labels to train the target model and the reference model.[1] Recall that several attacks require the knowledge of the target dataset's partial graph. To simulate and fairly evaluate different attacks, we construct an *attack dataset* which contains node pairs and labels representing whether they are linked or not. Specifically, we first select all node pairs that are linked. Then, we randomly sample the same number of node pairs that are not linked. We note that such negative sampling approach follows the common practice in the literature of link prediction [2, 25, 69]. Furthermore, the main metric we use, i.e., AUC (introduced below), is insensitive to the class imbalance issue [2, 21, 47] contrary to accuracy. Next, we split the attack dataset randomly by half into *attack training dataset* and *attack testing dataset*.[2] We use the attack training dataset to train our attack models when the target dataset's partial graph is part of the adversary's knowledge. We use attack testing dataset to evaluate all our attacks. For the attacks that have a shadow dataset, we also construct an attack dataset on the shadow dataset to train the attack model. Note that we do not split this attack dataset because we do not use it for evaluation.

[1]We do not train the reference model for attacks when $\mathcal{F}$ is unavailable.

[2]We perform additional experiments and observe that training set size does not have a strong impact on the attack performance, results are presented in Figure 7 in Appendix.

Metric. We use AUC (area under the ROC curve) as our main evaluation metric. AUC is frequently used in binary classification tasks [2, 21, 26, 32, 46, 47, 69], it is threshold independent. For convenience, we refer to node pairs that are linked as *positive node pairs* and those that are not linked as *negative node pairs*. If we rank node pairs according to the probability that there is a link between them, then AUC is the probability that a randomly selected positive node pair ranks higher than a randomly selected negative node pair. When performing random guessing, i.e., we rank all node pairs uniformly at random, the AUC value is 0.5. Note that we also calculate Precision and Recall for all supervised attacks (see Table 17, Table 18, Table 19, Table 20, Table 21, and Table 22 in Appendix).

Models. We use a graph convolutional network with 2 hidden layers for both the target model and the shadow target model, and assume they share the same architecture (see Section 3). Note that we also evaluate the scenario where the target model and the shadow model have different architectures later in this section and find the performances of our attacks are similar. The number of neurons in the hidden layer is set to 16. We adopt the frequently used ReLU and softmax as activation functions for the first hidden layer and the second hidden layer, respectively. Note that we append Dropout (the rate is 0.5) to the output of the hidden layer to prevent overfitting. We train 100 epochs with a learning rate of 0.01. Cross-entropy is adopted as the loss function and we use the Adam optimizer to update the model parameters. Our GNNs are implemented based on publicly available code.[3] Experimental results show that our GNNs achieve similar performance as reported in other papers. We omit them to preserve space.

We use an MLP with 2 hidden layers as the reference model and the shadow reference model. Hyperparameters, including the number of neurons in the hidden layer, activation functions, loss function, optimizer, epochs, and learning rate are the same as those of the target model.

We use an MLP with 3 hidden layers as our attack model. The number of neurons for all hidden layers is 32. ReLU is adopted as the activation function for hidden layers and softmax is used as the output activation function. We append Dropout (the rate is 0.5) to each hidden layer to prevent overfitting. We train 50 epochs with a learning rate of 0.001. The loss function is cross-entropy and the optimizer is Adam.

We run all experiments with this setting for 5 times and report the average value and the standard deviation of AUC scores. Note that for Attack-0 and Attack-2, the AUC scores keep the same since these two attacks are unsupervised.

5.2 Attack Performance

Attack-0: $\mathcal{K} = (\times, \times, \times)$. In this attack, the adversary only relies on measuring the distance of two nodes' posteriors obtained from the target model. We compare 8 different distance metrics and Figure 1 shows the results. First, we observe that Correlation distance achieves the best performance followed by Cosine distance across all datasets. In contrast, Canberra distance performs the worst. For instance, on the Citeseer dataset, the AUC scores for Correlation distance and Cosine distance are 0.959 and 0.946, respectively, while the AUC score for Canberra distance is 0.801. Note that both Correlation distance and Cosine distance measure the inner product between two vectors, or the "angle" of two vectors while other distance metrics do not. Second, we find that the performance of the same metric on different datasets is different. For instance, the AUC of Correlation distance on Citeseer is 0.959 compared to 0.635 on ENZYMES.

As mentioned in Section 4, unsupervised attacks could not provide a concrete prediction. To tackle this, we propose to use clustering, such as K-means. Concretely, we obtain a set of node pairs' distances, and perform K-means on these distances with K being set to 2. The cluster with lower (higher) average distance value is considered as the set of positive (negative) node pairs. Our experiments show that this method is effective. For instance, on the Citeseer dataset, we obtain 0.788 Precision, 0.991 Recall, and 0.878 F1-Score. The complete results are summarized in Table 15 in Appendix. Another method we could use is to assume that the adversary has a certain number of labeled edges, either from the target dataset or the shadow dataset. The former follows the same setting as our Attack-3, Attack-4, Attack-6, and Attack-7, and the latter is equivalent to Attack-1 and Attack-5. The corresponding results will be shown later.

Figure 2 shows the frequency of Correlation distance computed on posteriors obtained from the target model for both positive node pairs and negative node pairs in attack testing datasets. The x-axis is the value of Correlation distance and the y-axis is the number of pairs. A clear trend is that for all

[3] https://github.com/tkipf/gcn

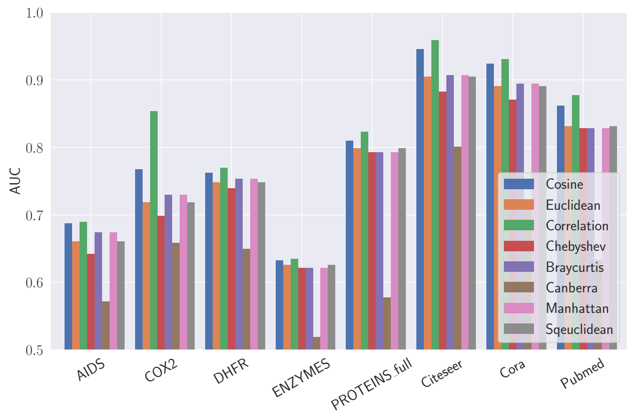


Figure 1: AUC for Attack-0 on all the 8 datasets with all the 8 distance metrics. The x-axis represents the dataset and the y-axis represents the AUC score.

Table 4: Average AUC with standard deviation for Attack-1 on all the 8 datasets. Best results are highlighted in bold.

	Shadow Dataset							
Target Dataset	AIDS	COX2	DHFR	ENZYMES	PROTEINS_full	Citeseer	Cora	Pubmed
AIDS	-	0.720 ± 0.009	0.690 ± 0.005	**0.730 ± 0.010**	0.720 ± 0.005	0.689 ± 0.019	0.650 ± 0.025	0.667 ± 0.014
COX2	0.755 ± 0.032	-	0.831 ± 0.005	0.739 ± 0.116	**0.832 ± 0.009**	0.762 ± 0.009	0.773 ± 0.008	0.722 ± 0.024
DHFR	0.689 ± 0.004	**0.771 ± 0.004**	-	0.577 ± 0.044	0.701 ± 0.010	0.736 ± 0.005	0.740 ± 0.003	0.663 ± 0.010
ENZYMES	**0.747 ± 0.014**	0.695 ± 0.023	0.514 ± 0.041	-	0.691 ± 0.030	0.680 ± 0.012	0.663 ± 0.009	0.637 ± 0.018
PROTEINS_full	0.775 ± 0.020	0.821 ± 0.016	0.528 ± 0.038	0.822 ± 0.020	-	**0.823 ± 0.004**	0.809 ± 0.015	0.809 ± 0.013
Citeseer	0.801 ± 0.040	0.920 ± 0.006	0.842 ± 0.036	0.846 ± 0.042	0.848 ± 0.015	-	**0.965 ± 0.001**	0.942 ± 0.003
Cora	0.791 ± 0.019	0.884 ± 0.005	0.811 ± 0.024	0.804 ± 0.048	0.869 ± 0.012	**0.942 ± 0.001**	-	0.917 ± 0.002
Pubmed	0.705 ± 0.039	0.796 ± 0.007	0.704 ± 0.042	0.708 ± 0.067	0.752 ± 0.014	0.883 ± 0.006	**0.885 ± 0.005**	-

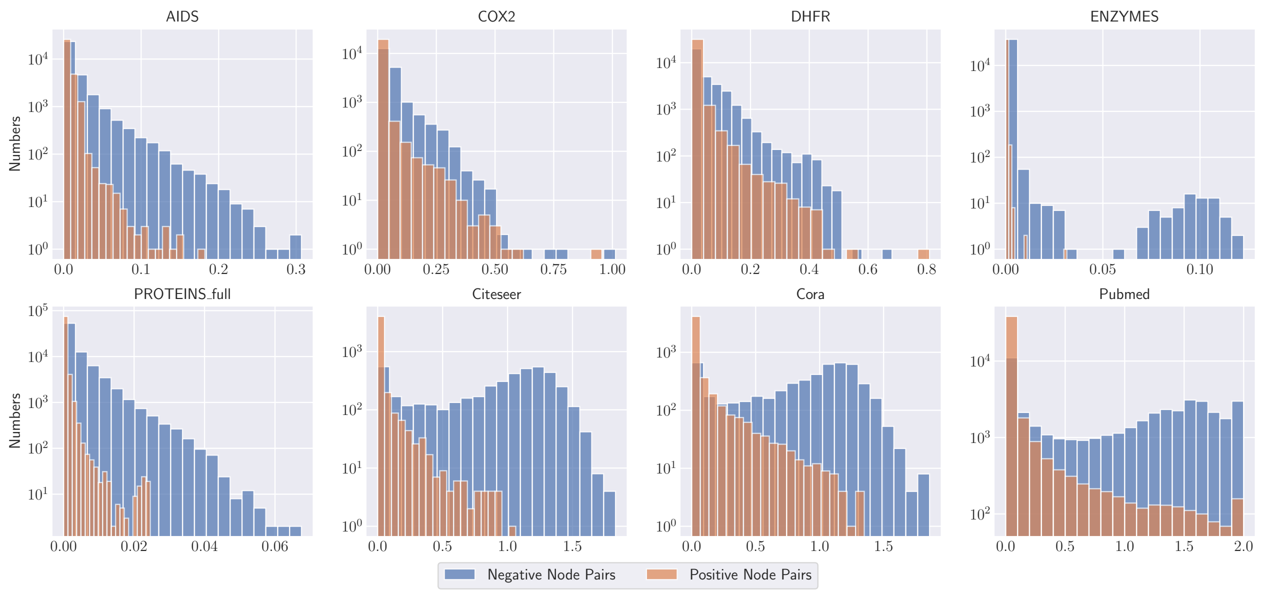


Figure 2: The Correlation distance distribution between nodes' posteriors for positive node pairs and negative node pairs on all the 8 datasets. The x-axis represents Correlation distance and the y-axis represents the number of node pairs.

datasets, the Correlation distance for positive node pairs is much smaller than negative node pairs. We select the top 50% of node pairs with lowest Correlation distance, group them, and calculate the AUC for each group. Due to the space limit, we only show the result on Pubmed (Table 5). We can see that the AUC drops when the Correlation distance increase, which indicates that Attack-0 works better on node pairs with lower Correlation distance. In general, the posteriors for positive node pairs are "closer" than that for negative node pairs. This verifies our intuition in Section 4: GNN can be considered as an aggregation function over the neighborhoods, if two nodes are linked, they aggregate with each other's features and therefore become closer.

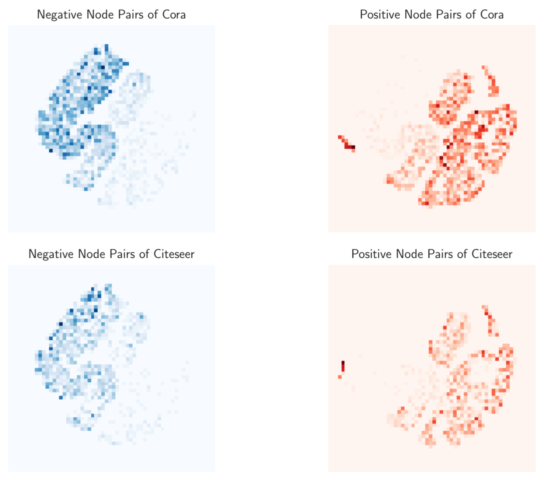
Attack-1: $\mathcal{K} = (\times, \times, \mathcal{D}')$**.** In this attack, the adversary can leverage a shadow dataset. In particular, for each dataset, we use one of the remaining datasets as the shadow dataset to perform the attack. Table 4 summarizes the results. We leave the blank in the diagonal because we do not use the target dataset itself as its shadow dataset.

Table 5: AUC in different Correlation distance levels for Attack-0 on Pubmed.

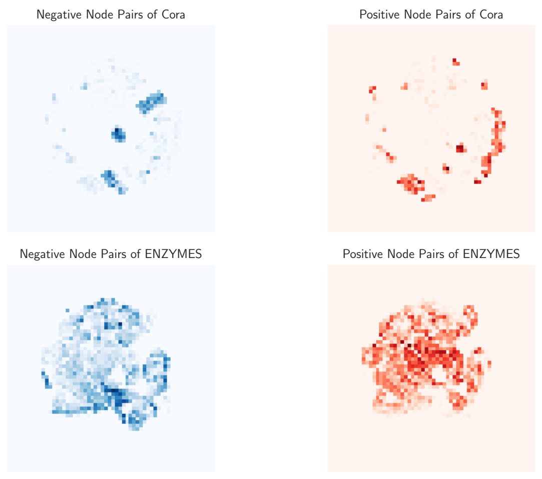
Correlation Distance	AUC	Correlation Distance	AUC
0.00-0.01	0.608	0.02-0.03	0.407
0.01-0.02	0.535	0.03-0.04	0.399

As we can see from Table 4, the AUC scores from the best-performing shadow dataset have a consistent improvement on almost all datasets compared to Attack-0. One exception is the COX2 dataset in which the AUC score decreases by 0.02. The results indicate that the adversary can indeed transfer the knowledge from the shadow dataset to enhance her attack.

An interesting finding is that for a chemical dataset, the best shadow dataset is normally a chemical dataset as well. Similar results can be observed for citation datasets. This shows that it is more effective to transfer knowledge across datasets from the same domain. To better understand this, we extract



(a)



(b)

Figure 3: The last hidden layer's output from the attack model of Attack-1 for 200 randomly sampled positive node pairs and 200 randomly sampled negative node pairs projected into a 2-dimension space using t-SNE. (a) Cora as the shadow dataset and Citeseer as the target dataset, (b) Cora as the shadow dataset and ENZYMES as the target dataset.

the attack model's last hidden layer's output (32-dimension) for positive node pairs and negative node pairs and project them into a 2-dimension space using t-Distributed Stochastic Neighbor Embedding (t-SNE) [61]. Figure 3a shows the results for Citeseer when using Cora as the shadow dataset, both of which are citation datasets. We can see that the positive (negative) node pairs from both the target dataset and the shadow dataset can be clustered into similar position, which indicates the positive (negative) node pairs from both datasets have similar distributions. This means if the attack model learns a decision boundary to separate positive nodes pairs from the negative node pairs on the shadow dataset, this decision boundary can be easily carried over to the target dataset.

In contrast, Figure 3b shows the results for ENZYMES (a chemical dataset) when using Cora (a citation dataset) as the shadow dataset. We see that the positive (negative) node pairs from the shadow dataset and the target dataset are distributed differently in the 2-dimension space. For example, the positive node pairs for Cora are clustered into the outer space of the circle area whereas the positive node pairs for ENZYMES are clustered into the inner space of the circle area. Therefore, it is hard for the adversary to perform an effective transferring attack. The underlying reason for this to happen is that graphs from the same domain have analogous graph structures and similar features. This leads to less information loss for our transferring attack.

Attack-2: $\mathcal{K} = (\mathcal{F}, \times, \times)$**.** In Attack-2, the adversary has the knowledge of the target dataset's nodes' attributes. As discussed in Section 4, she trains a reference model g by herself from $\mathcal{F}$. We compare four types of information mentioned in Section 4, and the results are shown in Figure 4. Note that

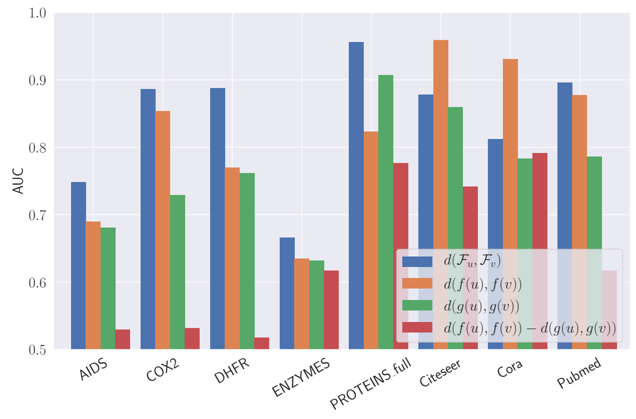


Figure 4: Average AUC for Attack-2 on all the 8 datasets with all the 4 types of information considered. The x-axis represents the dataset and the y-axis represents the AUC score.

we only show the results calculated with Correlation distance out of the 8 distance metrics (Table 13) since Correlation distance achieves the best performance in almost all settings. We can see that in all chemical datasets and one citation dataset, using the distance of target dataset's nodes' attributes leads to the best performance. For the other two citation datasets, using the distance between posteriors of the target model can get better performance. Nodes' attributes' dimensions are higher in citation datasets than in chemical datasets. In other words, the node attributes for citation datasets are sparser. For instance, we observe that most attributes are 0 in citation datasets. Therefore, we conclude that the attack can get

Table 6: Average AUC with standard deviation for Attack-3 on all the 8 datasets.

Dataset	AUC	Dataset	AUC
AIDS	0.961 ± 0.001	PROTEINS_full	0.958 ± 0.000
COX2	0.939 ± 0.002	Citeseer	0.973 ± 0.000
DHFR	0.934 ± 0.001	Cora	0.954 ± 0.001
ENZYMES	0.882 ± 0.001	Pubmed	0.947 ± 0.001

better performance using the Correlation distance between posteriors of the target model when the target dataset's nodes' attributes are in high dimension.

Attack-3: $\mathcal{K} = (\times, \mathcal{A}^*, \times)$**.** Table 6 shows the results for this attack. With the knowledge of the target dataset's partial graph, the average AUC score for all cases is over 0.9. Compared to Attack-2, the AUC scores on chemical datasets have an improvement over 10% and the AUC scores on citation datasets have an improvement over 2%.[4]

Compared to Attack-1 and Attack-2, Attack-3 achieves the best performance, this indicates the target dataset's partial graph is the most important component for an adversary for performing a link stealing attack. The reason is that the partial graph contains the ground truth links in the target dataset, which can be directly exploited by the attack model.

We further investigate the contribution of each feature set to the final prediction following the methodology of Dong et al. [16]. Concretely, when studying one feature set, we set other features' value to 0. As shown in Figure 5, the features extracted by applying pairwise operation over posteriors are most useful for the final prediction, followed by the features based on posteriors with different distance metrics. We note that our attack also achieves over 0.70 AUC on average when only using pairwise operation over entropy of posteriors as features. Moreover, our attack achieves the best performance when taking all the three feature sets together, which implies the combination of different features indeed improves the overall performance.

Attack-4: $\mathcal{K} = (\times, \mathcal{A}^*, \mathcal{D}')$**.** Table 7 shows the results for Attack-4. First, compared to Attack-1 ($\mathcal{K} = (\times, \times, \mathcal{D}')$), the overall performance of Attack-4 improves with the help of target dataset's partial graph $\mathcal{A}^*$. This is reasonable since the target dataset's partial graph contains some ground truth links from the target dataset. Second, we note that the performances of Attack-4 are worse than Attack-3 ($\mathcal{K} = (\times, \mathcal{A}^*, \times)$). Intuitively, the performance should be better since Attack-4 has more background knowledge. The reason for the performance degradation is that we do not take the pairwise vector operation (Table 14) over posteriors as the input for Attack-4 since we want to learn information from both the target dataset and the shadow dataset, and need to eliminate the dimension mismatch issue (as discussed in Section 4). Moreover, the

[4]Attack-2 achieves relatively high AUC scores on citation datasets.

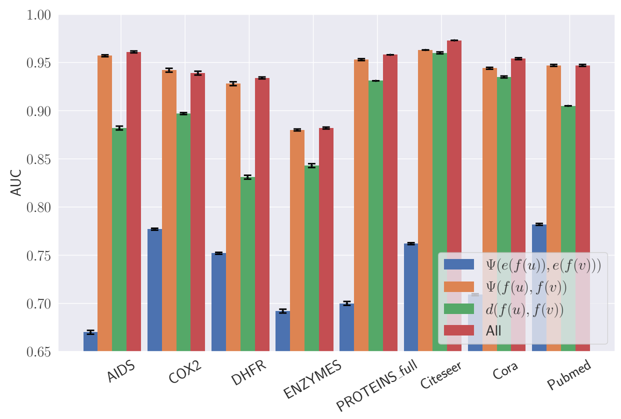


Figure 5: Average AUC for Attack-3 on all the 8 datasets with different set of features. The x-axis represents the dataset and the y-axis represents the AUC score.

results also indicate that compared to the shadow dataset, the target dataset's partial graph is more informative.

Attack-5: $\mathcal{K} = (\mathcal{F}, \times, \mathcal{D}')$**.** In Attack-5, the adversary has the knowledge of target dataset's nodes' attributes as well as a shadow dataset, evaluation results are shown in Table 8. We observe that Attack-5 performs better than both Attack-1 (only with $\mathcal{D}'$) and Attack-2 (only with $\mathcal{F}$). This shows the combination of $\mathcal{F}$ and $\mathcal{D}'$ can lead to a better link stealing performance. Furthermore, we observe similar trends as for Attack-1, that is the attack performs better if the shadow dataset comes from the same domain as the target dataset.

Attack-6: $\mathcal{K} = (\mathcal{F}, \mathcal{A}^*, \times)$**.** The result of Attack-6 on all datasets is shown in Table 10. We can see that for almost all datasets (except ENZYMES), the AUC scores are over 0.95, which means this attack achieves an excellent performance. In particular, the AUC score is nearly 1 on PROTEINS_full. Moreover, Attack-6 consistently outperforms Attack-2 ($\mathcal{K} = (\mathcal{F}, \times, \times)$). This further validates the effectiveness of $\mathcal{A}^*$ in helping the adversary to infer links. Another finding is that for chemical datasets, the information of target dataset's partial graph brings a larger improvement than the citation datasets. One possible explanation is that the nodes' attributes in chemical datasets contain less information (they are in lower dimension), thus the target dataset's partial graph contributes more to the final prediction performance.

Attack-7: $\mathcal{K} = (\mathcal{F}, \mathcal{A}^*, \mathcal{D}')$**.** The results of Attack-7 are summarized in Table 9. Compared to Attack-5 ($\mathcal{K} = (\mathcal{F}, \times, \mathcal{D}')$), the overall performances improve with the help of $\mathcal{A}^*$. We would expect the adversary's accuracy is better than that of Attack-6 ($\mathcal{K} = (\mathcal{F}, \mathcal{A}^*, \times)$) since she has more background knowledge. However, we observe that the performance drops from Attack-6 to Attack-7. We suspect this is due to the fact that we want to learn information from both the target dataset and the shadow dataset, to avoid the dimension mismatch

Table 7: Average AUC with standard deviation for Attack-4 on all the 8 datasets. Best results are highlighted in bold.

	Shadow Dataset							
Target Dataset	AIDS	COX2	DHFR	ENZYMES	PROTEINS_full	Citeseer	Cora	Pubmed
AIDS	-	0.750 ± 0.009	**0.763** ± **0.010**	0.733 ± 0.007	0.557 ± 0.009	0.729 ± 0.015	0.702 ± 0.010	0.673 ± 0.009
COX2	0.802 ± 0.031	-	**0.866** ± **0.004**	0.782 ± 0.012	0.561 ± 0.030	0.860 ± 0.002	0.853 ± 0.004	0.767 ± 0.023
DHFR	0.758 ± 0.022	**0.812** ± **0.005**	-	0.662 ± 0.030	0.578 ± 0.067	0.799 ± 0.002	0.798 ± 0.009	0.736 ± 0.005
ENZYMES	**0.741** ± **0.010**	0.684 ± 0.024	0.670 ± 0.008	-	0.733 ± 0.019	0.624 ± 0.002	0.627 ± 0.014	0.691 ± 0.012
PROTEINS_full	0.715 ± 0.009	0.802 ± 0.025	0.725 ± 0.041	0.863 ± 0.010	-	0.784 ± 0.031	0.815 ± 0.012	**0.867** ± **0.003**
Citeseer	0.832 ± 0.078	0.940 ± 0.005	0.914 ± 0.007	0.879 ± 0.062	0.833 ± 0.088	-	**0.967** ± **0.001**	0.955 ± 0.003
Cora	0.572 ± 0.188	0.899 ± 0.003	0.887 ± 0.014	0.878 ± 0.045	0.738 ± 0.168	**0.945** ± **0.001**	-	0.924 ± 0.005
Pubmed	0.777 ± 0.056	0.893 ± 0.001	0.90 ± 0.006	0.866 ± 0.002	0.806 ± 0.042	**0.907** ± **0.004**	0.902 ± 0.001	-

Table 8: Average AUC with standard deviation for Attack-5 on all the 8 datasets. Best results are highlighted in bold.

	Shadow Dataset							
Target Dataset	AIDS	COX2	DHFR	ENZYMES	PROTEINS_full	Citeseer	Cora	Pubmed
AIDS	-	0.841 ± 0.003	0.846 ± 0.009	0.795 ± 0.016	**0.875** ± **0.002**	0.839 ± 0.006	0.793 ± 0.015	0.787 ± 0.008
COX2	0.832 ± 0.036	-	**0.977** ± **0.002**	0.874 ± 0.020	0.946 ± 0.003	0.911 ± 0.004	0.908 ± 0.004	0.887 ± 0.004
DHFR	0.840 ± 0.018	**0.988** ± **0.001**	-	0.757 ± 0.032	0.970 ± 0.004	0.909 ± 0.010	0.911 ± 0.009	0.860 ± 0.004
ENZYMES	0.639 ± 0.005	0.581 ± 0.010	0.587 ± 0.005	-	0.608 ± 0.001	**0.685** ± **0.005**	0.674 ± 0.007	0.663 ± 0.002
PROTEINS_full	0.948 ± 0.007	**0.981** ± **0.004**	0.968 ± 0.014	0.818 ± 0.017	-	0.970 ± 0.002	0.876 ± 0.010	0.885 ± 0.003
Citeseer	0.773 ± 0.048	0.666 ± 0.018	0.652 ± 0.020	0.860 ± 0.049	0.794 ± 0.009	-	**0.969** ± **0.002**	0.967 ± 0.001
Cora	0.743 ± 0.017	0.587 ± 0.012	0.568 ± 0.009	0.778 ± 0.052	0.686 ± 0.018	**0.956** ± **0.001**	-	0.936 ± 0.002
Pubmed	0.777 ± 0.030	0.661 ± 0.018	0.645 ± 0.008	0.786 ± 0.041	0.741 ± 0.008	0.938 ± 0.007	**0.941** ± **0.007**	-

Table 9: Average AUC with standard deviation for Attack-7 on all the 8 datasets. Best results are highlighted in bold.

	Shadow Dataset							
Target Dataset	AIDS	COX2	DHFR	ENZYMES	PROTEINS_full	Citeseer	Cora	Pubmed
AIDS	-	**0.925** ± **0.001**	0.913 ± 0.005	0.784 ± 0.010	0.848 ± 0.010	0.538 ± 0.022	0.520 ± 0.011	0.849 ± 0.004
COX2	0.954 ± 0.007	-	**0.982** ± **0.001**	0.874 ± 0.010	0.898 ± 0.030	0.947 ± 0.003	0.940 ± 0.007	0.875 ± 0.034
DHFR	0.982 ± 0.002	**0.992** ± **0.00**	-	0.871 ± 0.017	0.966 ± 0.008	0.933 ± 0.008	0.947 ± 0.012	0.937 ± 0.003
ENZYMES	**0.698** ± **0.007**	0.691 ± 0.008	0.671 ± 0.003	-	0.610 ± 0.001	0.657 ± 0.009	0.662 ± 0.006	0.677 ± 0.001
PROTEINS_full	0.984 ± 0.002	0.962 ± 0.010	0.986 ± 0.002	**0.993** ± **0.001**	-	0.840 ± 0.013	0.823 ± 0.006	0.987 ± 0.005
Citeseer	0.816 ± 0.048	0.791 ± 0.033	0.702 ± 0.025	0.880 ± 0.057	0.902 ± 0.026	-	**0.977** ± **0.000**	0.964 ± 0.000
Cora	0.746 ± 0.068	0.680 ± 0.038	0.574 ± 0.038	0.888 ± 0.014	0.695 ± 0.10	**0.960** ± **0.001**	-	0.935 ± 0.001
Pubmed	0.807 ± 0.016	0.712 ± 0.025	0.710 ± 0.006	0.881 ± 0.009	0.739 ± 0.012	**0.956** ± **0.001**	0.949 ± 0.001	-

Table 10: Average AUC with standard deviation for Attack-6 on all the 8 datasets.

Dataset	AUC	Dataset	AUC
AIDS	0.979 ± 0.001	PROTEINS_full	0.999 ± 0.000
COX2	0.987 ± 0.001	Citeseer	0.981 ± 0.000
DHFR	0.992 ± 0.001	Cora	0.964 ± 0.000
ENZYMES	0.891 ± 0.001	Pubmed	0.970 ± 0.000

problem, Attack-7 uses fewer features than Attack-6 (similar to the reason that Attack-4 performs worse than Attack-3).

Comparison with Link Prediction. We further compare all our attacks with a traditional link prediction method [40]. More specifically, we build an MLP with features summarized from the target model's partial graph, including Common neighbor, Jaccard index, and Preferential attachment [40]. As we can see from Figure 6, most of our attacks outperforms the link prediction method. For instance, on the COX2 dataset, all our 8 attacks outperform the link prediction model, the best attack (Attack-6) achieves more than 20% performance gain. This demonstrates that GNNs lead to more severe privacy risks than traditional link prediction.

Effect of Different GNN Structures. In our experiments, we adopt the same architecture for both the target model and the shadow target model by default for transferring attack scenarios. We further evaluate the impact of the shadow target model using different architectures. Note that for space reasons, we only report the results of Attack-1. Results for other attacks are similar. We set the number of hidden layers to 3 for the shadow target model (the target model has 2 hidden layers). The results are summarized in Table 16 in Appendix. We find the average AUC scores of our attack are maintained at the same level or even higher for certain datasets compared with the scenario where the shadow target model and the shadow model have the same architecture. For

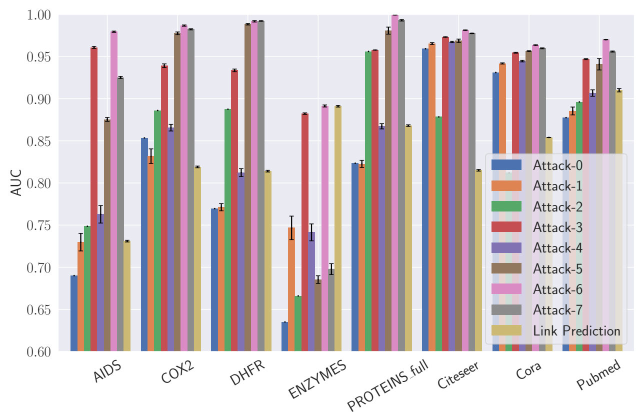


Figure 6: Average AUC with standard deviation for all the attacks on all the 8 datasets. For each attack, we list its best result. The x-axis represents the dataset and the y-axis represents the AUC score.

Table 11: Average AUC with standard deviation for Attack-6 when using GraphSAGE or GAT as the target model on all the 8 datasets.

Dataset	AUC (GraphSAGE)	AUC (GAT)
AIDS	0.977 ± 0.002	0.968 ± 0.001
COX2	0.982 ± 0.001	0.984 ± 0.001
DHFR	0.990 ± 0.001	0.995 ± 0.000
ENZYMES	0.747 ± 0.001	0.766 ± 0.004
PROTEINS_full	0.999 ± 0.000	0.999 ± 0.000
Citeseer	0.938 ± 0.000	0.972 ± 0.000
Cora	0.883 ± 0.001	0.958 ± 0.000
Pubmed	0.923 ± 0.000	0.965 ± 0.000

instance, on the Citeseer dataset, we obtain 0.924 AUC, while the original attack achieves 0.965. In other words, our attacks are still effective when the shadow target model and the target model have different architectures.

Attacks on Other GNNs. We further investigate whether our attacks are applicable to other GNN models besides GCN. Concretely, we focus on GraphSAGE [27] and GAT [62]. We implement GraphSAGE[5] and GAT[6] based on publicly available code and only report the results of Attack-6. Table 11 shows that our attack has similar AUC scores on GraphSAGE and GAT compared to GCN. For instance, on the COX2 dataset, our attack against GraphSAGE and GAT achieves AUC of 0.982 and 0.984, respectively (the corresponding AUC for GCN is 0.987). This further demonstrates that our attacks are generally applicable.

Possible Defense. We try to restrict the GNN model to output k largest posteriors as a defense mechanism to mitigate our attacks. The intuition is that the smaller k is, the less information the model reveals. Here, we fix $k = 2$ and report the results for Attack-3. Note that we have similar observations for other attacks. Experimental results in Table 12 show that this defense indeed reduces the performance of our attack. However, the performance drop is not very big, i.e., our attack still achieves relatively high AUC scores. For instance, on the Citeseer dataset, this defense reduces Attack-3's performance by less than 2%. On the AIDS dataset, the attack's performance drop is higher but AUC being 0.855 still indicates our attack is effective. We also note that the defense will impact the utility of the model. In other words, it is a trade-off between utility and privacy. In conclusion, the top-k defense is not effective enough to defend against our attacks.

[5] https://github.com/williamleif/GraphSAGE
[6] https://github.com/PetarV-/GAT

Table 12: Average AUC with standard deviation for Attack-3 when only reporting top-2 posteriors on all the 8 datasets.

Dataset	AUC	Dataset	AUC
AIDS	0.855 ± 0.004	PROTEINS_full	0.954 ± 0.001
COX2	0.839 ± 0.005	Citeseer	0.958 ± 0.000
DHFR	0.851 ± 0.003	Cora	0.945 ± 0.001
ENZYMES	0.876 ± 0.002	Pubmed	0.946 ± 0.001

We can also leverage differential privacy (DP) and adversarial examples to mitigate our attacks. In detail, we can adopt edge-DP developed for social networks [28, 68] to defend against our attacks. Borrowing the idea from previous work [31, 32], we can also add carefully crafted noise to the prediction of GNN to fool the adversary. We plan to explore both of them in the future.

Summary of Results. In summary, we have made the following observations from our experimental results.

- Our attacks can effectively steal the links from GNNs. For instance, our Attack-6 can achieve average AUC scores over 0.95 on 7 out of 8 datasets, which demonstrate that the GNNs are vulnerable to our attacks.
- Generally speaking, the performances of the attack are better if there is more background knowledge as shown in Figure 6. However, we find the impact of different knowledge is different. In particular, the target dataset's partial graph is the most informative. For instance, Attack-3 ($\mathcal{K} = (\times, \mathcal{A}^*, \times)$) significantly outperforms Attack-1 ($\mathcal{K} = (\times, \times, \mathcal{D}')$) and Attack-2 ($\mathcal{K} = (\mathcal{F}, \times, \times)$).
- Our transferring attack can achieve good performance. Furthermore, we find that our transferring attack achieves better performance when the shadow dataset and the target dataset are from the same domain as validated by experimental results for Attack-1 and Attack-5.

6 Related Work

Various research has shown that machine learning models are vulnerable to security and privacy attacks [9, 12, 30, 36–38, 49, 50, 53, 55, 60]. In this section, we mainly survey four of these attacks that are most relevant to ours.

Membership Inference. In membership inference attacks [6, 10, 29, 39, 42, 43, 54, 56, 58, 67], the adversary aims to infer whether a data sample is in the target model's training dataset or not. Shokri et al. [56] propose the first membership inference attacks against machine learning models and demonstrate its relationship with model overfitting. Salem et al. [54] further show membership inference attacks are broadly applicable at low cost via relaxing assumptions on the adversary. To mitigate attacks, many empirical defenses [32, 42, 54, 56] have been proposed. For instance, Nasr et al. [42] propose to mitigate attacks via formulating the defense as a min-max optimization problem which tries to decrease the accuracy loss and increase the membership privacy. Salem et al. [54] explore dropout and model stacking to mitigate membership inference attacks. More recently, Jia et al. [32] leverage adversarial examples to fool the adversary and show their defense has a formal utility guarantee. Other attacks in this space study membership inference in natural language processing models [57], generative models [8, 29], federated learning [41], and biomedical data [26].

Model Inversion. In model inversion attacks [20, 21, 30, 41, 48], the adversary aims to learn sensitive attributes of training data from target models. For example, Fredrikson et al. [21] propose the model inversion attack in which the adversary can infer the patient's genetic markers given the model and some demographic information about the patients. Fredrikson et al. [20] further explore the model inversion attacks on decision trees and neural networks via exploiting the confidence score values revealed along with predictions. Melis et al. [41] revealed that in the collaborative learning scenarios, when the target model updated with new training data, the adversary could infer sensitive attributes about the new training data.

Model Extraction. In model extraction attacks [7, 30, 60, 63], the adversary aims to steal the parameters of a certain target model or mimic its behaviors. Tramér et al. [60] show that an adversary can exactly recover the target model's parameters via solving the equations for certain models, e.g., linear models. Wang and Gong [63] propose attacks to steal the hyperparameters and show their attacks are broadly applicable to a variety of machine learning algorithms, e.g., ridge regression and SVM. Orekondy et al. [44] propose a functionality stealing attack aiming at mimicking the behaviors of the target model. Concretely, they query the target model and use the query-prediction pairs to train a "knockoff" model. Jagielski et al. [30] improve the query efficiency of learning-based model extraction attacks and develop the practical functionally-equivalent model whose predictions are identical to the target model on all inputs without training model's weights. Some defenses [34, 45] have been proposed to defend against model extraction attacks. For instance, Juuti et al. [34] propose to detect malicious queries via analyzing the distribution of consecutive API queries and raises an alarm when the distribution different from benign queries. Orekondy et al [45] propose a utility-constrained defense against neural network model stealing attacks via adding perturbations to the output of the target model.

Adversarial Attacks on Graph Neural Networks. Some recent studies [3, 13, 64, 66, 71, 73, 74] show that GNNs are vulnerable to adversarial attacks. In particular, the adversary can fool GNNs via manipulating the graph structure and/or node features. For instance, Zügner et al. [73] introduce adversarial attacks to attributed graphs and focus on both training and testing phase. In particular, their attacks target both node's features and graph structure and show that the node classification accuracy drops with a few perturbations. Bojchevski et al. [3] analyze the vulnerability of node embeddings to graph structure perturbation via solving a bi-level optimization problem based on eigenvalue perturbation theory. Zügner and Günnemann [74] investigate training time attacks on GNNs for node classification via treating the graph as a hyperparameter to optimize. Wang and Gong [64] propose an attack to evade the collective classification based classifier via perturbing the graph structure, which can also transfer to GNNs. Dai et al. [13] propose to fool the GNNs via manipulating the combinatorial structure of data and try to learn generalizable attack policy via reinforcement learning. Zhang et al. [71] propose a subgraph based backdoor attack to GNN based graph classification. In particular, a GNN classifier outputs a target label specified by an adversary when a predefined subgraph is injected to the testing graph. These studies are different from our work since we aim to steal links from GNNs.

To mitigate attacks, many defenses [4, 66, 72, 75] have been proposed. For instance, Zhu et al. [72] propose to enhance the robustness of GCNs via using Gaussian distributions in graph convolutional layers to mitigate the effects of adversarial attacks and leveraged attention mechanism to impede the propagation of attacks. Zügner and Günnemann [75] propose a learning principle that improves the robustness of the GNNs and show provable robustness guarantees against nodes' attributes perturbation. Bojchevski et al. [3] propose to certify the robustness against graph structure perturbation for a general class of models, e.g., GNNs, via exploiting connections to PageRank and Markov decision processes. These defenses are designed to improve the robustness of GNNs rather than preventing the privacy leakage of it. Note that there are also some attacks and defenses on graph that focus on non-GNN models [11, 33]. For instance, Chen et al. [11] propose attacks that mislead the behavior of graph-cluster algorithm and show some practical defenses. Jia et al. [33] propose certified defense which is based on randomized smoothing to defend against adversarial structural attacks to community detection.

7 Conclusion and Future Work

In this paper, we propose the first link stealing attacks against GNNs. Specifically, we show that, given a black-box access to a target GNN model, an adversary can accurately infer whether there exists a link between any pair of nodes in a graph that is used to train the GNN model. We propose a threat model to systematically characterize an adversary's background knowledge along three dimensions. By jointly considering the three dimensions, we define 8 link stealing attacks and propose novel methods to realize them. Extensive evaluation over 8 real-world datasets shows that our attacks can accurately steal links. Interesting future work includes generalizing our attacks to GNNs for graph classification and defending against our attacks.

Acknowledgments

We thank the anonymous reviewers and our shepherd Minhui Xue for constructive feedback. This work is partially funded by the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (funding number ZT-I-OO1 4) and National Science Foundation grant No. 1937787.

References

[1] James Atwood and Don Towsley. Diffusion-Convolutional Neural Networks. In *NIPS*, pages 1993–2001, 2016.

[2] Michael Backes, Mathias Humbert, Jun Pang, and Yang Zhang. walk2friends: Inferring Social Links from Mobility Profiles. In *CCS*, pages 1943–1957, 2017.

[3] Aleksandar Bojchevski and Stephan Günnemann. Adversarial Attacks on Node Embeddings via Graph Poisoning. In *ICML*, pages 695–704, 2019.

[4] Aleksandar Bojchevski and Stephan Günnemann. Certifiable Robustness to Graph Perturbations. In *NeurIPS*, pages 8317–8328, 2019.

[5] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönauer, S. V. N. Vishwanathan, Alexander J. Smola, and Hans-Peter Kriegel. Protein Function Prediction via Graph Kernels. *Bioinformatics*, 2005.

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

[7] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. Exploring Connections Between Active Learning and Model Extraction. In *USENIX Security*, 2020.

[8] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. GAN-Leaks: A Taxonomy of Membership Inference Attacks against GANs. In *CCS*, 2020.

[9] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When Machine Unlearning Jeopardizes Privacy. *CoRR abs/2005.02205*, 2020.

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

[11] Yizheng Chen, Yacin Nadji, Athanasios Kountouras, Fabian Monrose, Roberto Perdisci, Manos Antonakakis, and Nikolaos Vasiloglou. Practical Attacks Against Graph-based Clustering. In *CCS*, pages 1125–1142, 2017.

[12] Yizheng Chen, Shiqi Wang, Dongdong She, and Suman Jana. On Training Robust PDF Malware Classifiers. In *USENIX Security*, 2020.

[13] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial Attack on Graph Structured Data. In *ICML*, pages 1123–1132, 2018.

[14] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*, pages 3837–3845, 2016.

[15] Paul D. Dobson and Andrew J. Doig. Distinguishing Enzyme Structures from Non-Enzymes without Alignments. *Journal of Molecular Biology*, 2003.

[16] Yuxiao Dong, Reid A. Johnson, and Nitesh V. Chawla. Will This Paper Increase Your *h*-index?: Scientific Impact Prediction. In *WSDM*, pages 149–158, 2015.

[17] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking Graph Neural Networks. *CoRR abs/2003.00982*, 2020.

[18] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A Fair Comparison of Graph Neural Networks for Graph Classification. In *ICLR*, 2020.

[19] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. Graph Neural Networks for Social Recommendation. In *WWW*, pages 417–426, 2019.

[20] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *CCS*, pages 1322–1333, 2015.

[21] Matt Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. In *USENIX Security*, pages 17–32, 2014.

[22] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *ICML*, pages 1263–1272, 2017.

[23] Neil Zhenqiang Gong and Bin Liu. You are Who You Know and How You Behave: Attribute Inference Attacks via Users' Social Friends and Behaviors. In *USENIX Security*, pages 979–995, 2016.

[24] Neil Zhenqiang Gong, Ameet Talwalkar, Lester W. Mackey, Ling Huang, Eui Chul Richard Shin, Emil Stefanov, Elaine Shi, and Dawn Song. Joint Link Prediction and Attribute Inference Using a Social-Attribute Network. *ACM Transactions on Intelligent Systems and Technology*, 2014.

[25] Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. In *KDD*, pages 855–864, 2016.

[26] Inken Hagestedt, Yang Zhang, Mathias Humbert, Pascal Berrang, Haixu Tang, XiaoFeng Wang, and Michael Backes. MBeacon: Privacy-Preserving Beacons for DNA Methylation Data. In *NDSS*, 2019.

[27] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, pages 1025–1035, 2017.

[28] Michael Hay, Chao Li, Gerome Miklau, and David D. Jensen. Accurate Estimation of the Degree Distribution of Private Networks. In *ICDM*, pages 169–178, 2009.

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

[30] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. High Accuracy and High Fidelity Extraction of Neural Networks. In *USENIX Security*, 2020.

[31] Jinyuan Jia and Neil Zhenqiang Gong. AttriGuard: A Practical Defense Against Attribute Inference Attacks via Adversarial Machine Learning. In *USENIX Security*, pages 513–529, 2018.

[32] Jinyuan Jia, Ahmed Salem, Michael Backes, Yang Zhang, and Neil Zhenqiang Gong. MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples. In *CCS*, pages 259–274, 2019.

[33] Jinyuan Jia, Binghui Wang, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified Robustness of Community Detection against Adversarial Structural Perturbation via Randomized Smoothing. In *WWW*, pages 2718–2724, 2020.

[34] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N. Asokan. PRADA: Protecting Against DNN Model Stealing Attacks. In *Euro S&P*, pages 512–527, 2019.

[35] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.

[36] Klas Leino and Matt Fredrikson. Stolen Memories: Leveraging Model Memorization for Calibrated White-Box Membership Inference. In *USENIX Security*, 2020.

[37] Shaofeng Li, Shiqing Ma, Minhui Xue, and Benjamin Zi Hao Zhao. Deep Learning Backdoors. *CoRR abs/2007.08273*, 2020.

[38] Zheng Li, Chengyu Hu, Yang Zhang, and Shanqing Guo. How to Prove Your Model Belongs to You: A Blind-Watermark based Framework to Protect Intellectual Property of DNN. In *ACSAC*, pages 126–137, 2019.

[39] Zheng Li and Yang Zhang. Label-Leaks: Membership Inference Attack with Label. *CoRR abs/2007.15528*, 2020.

[40] David Liben-Nowell and Jon Kleinberg. The Link-prediction Problem for Social Networks. *Journal of the American Society for Information Science and Technology*, 2007.

[41] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Exploiting Unintended Feature Leakage in Collaborative Learning. In *S&P*, pages 497–512, 2019.

[42] Milad Nasr, Reza Shokri, and Amir Houmansadr. Machine Learning with Membership Privacy using Adversarial Regularization. In *CCS*, pages 634–646, 2018.

[43] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning. In *S&P*, pages 1021–1035, 2019.

[44] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff Nets: Stealing Functionality of Black-Box Models. In *CVPR*, pages 4954–4963, 2019.

[45] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Prediction Poisoning: Towards Defenses Against DNN Model Stealing Attacks. In *ICLR*, 2020.

[46] Jun Pang and Yang Zhang. DeepCity: A Feature Learning Framework for Mining Location Check-Ins. In *ICWSM*, pages 652–655, 2017.

[47] Jun Pang and Yang Zhang. Quantifying Location Sociality. In *HT*, pages 145–154, 2017.

[48] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael Wellman. SoK: Towards the Science of Security and Privacy in Machine Learning. In *Euro S&P*, pages 399–414, 2018.

[49] Nicolas Papernot, Patrick D. McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical Black-Box Attacks Against Machine Learning. In *ASIACCS*, pages 506–519, 2017.

[50] Erwin Quiring, Alwin Maier, and Konrad Rieck. Misleading Authorship Attribution of Source Code using Adversarial Learning. In *USENIX Security*, pages 479–496, 2019.

[51] Kaspar Riesen and Horst Bunke. *Structural, Syntactic, and Statistical Pattern Recognition*. Springer, 2008.

[52] Ahmed Salem, Apratim Bhattacharya, Michael Backes, Mario Fritz, and Yang Zhang. Updates-Leak: Data Set Inference and Reconstruction Attacks in Online Learning. In *USENIX Security*, pages 1291–1308, 2020.

[53] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. Dynamic Backdoor Attacks Against Machine Learning Models. *CoRR abs/2003.03675*, 2020.

[54] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. ML-Leaks: Model and Data Independent Membership Inference Attacks and Defenses on Machine Learning Models. In *NDSS*, 2019.

[55] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *NeurIPS*, pages 6103–6113, 2018.

[56] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership Inference Attacks Against Machine Learning Models. In *S&P*, pages 3–18, 2017.

[57] Congzheng Song and Vitaly Shmatikov. Auditing Data Provenance in Text-Generation Models. In *KDD*, pages 196–206, 2019.

[58] Congzheng Song and Reza Shokri. Robust Membership Encoding: Inference Attacks and Copyright Protection for Deep Learning. In *ASIACCS*, 2020.

[59] Jeffrey Sutherland, Lee O'Brien, and Donald Weaver. SplineFitting with a Genetic Algorithm: A Method for Developing Classification Structure Activity Relationships. *Journal of Chemical Information and Computer Sciences*, 2003.

[60] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *USENIX Security*, pages 601–618, 2016.

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

[62] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, 2018.

[63] Binghui Wang and Neil Zhenqiang Gong. Stealing Hyperparameters in Machine Learning. In *S&P*, pages 36–52, 2018.

[64] Binghui Wang and Neil Zhenqiang Gong. Attacking Graph-based Classification via Manipulating the Graph Structure. In *CCS*, pages 2023–2040, 2019.

[65] Binghui Wang, Jinyuan Jia, and Neil Zhenqiang Gong. Graph-based Security and Privacy Analytics via Collective Classification with Joint Weight Learning and Propagation. In *NDSS*, 2019.

[66] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial Examples for Graph Data: Deep Insights into Attack and Defense. In *IJCAI*, pages 4816–4823, 2019.

[67] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting. In *CSF*, pages 268–282, 2018.

[68] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Private Release of Graph Statistics using Ladder Functions. In *SIGMOD*, pages 731–745, 2015.

[69] Yang Zhang. Language in Our Time: An Empirical Analysis of Hashtags. In *WWW*, pages 2378–2389, 2019.

[70] Yang Zhang, Mathias Humbert, Bartlomiej Surma, Praveen Manoharan, Jilles Vreeken, and Michael Backes. Towards Plausible Graph Anonymization. In *NDSS*, 2020.

[71] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Backdoor Attacks to Graph Neural Networks. *CoRR abs/2006.11165*, 2020.

[72] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust Graph Convolutional Networks Against Adversarial Attacks. In *KDD*, pages 1399–1407, 2019.

[73] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial Attacks on Neural Networks for Graph Data. In *KDD*, pages 2847–2856, 2018.

[74] Daniel Zügner and Stephan Günnemann. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *ICLR*, 2019.

[75] Daniel Zügner and Stephan Günnemann. Certifiable Robustness and Robust Training for Graph Convolutional Networks. In *KDD*, pages 246–256, 2019.

A Appendix

Table 13: Distance metrics, $f_i(u)$ represents the i-th component of $f(u)$. Note that these metrics can be applied to nodes' attributes as well.

Metrics	Definition
Cosine	$1-\frac{f(u)\cdot f(v)}{\lVert f(u)\rVert_2 \lVert f(v)\rVert_2}$
Euclidean	$\lVert f(u)-f(v)\rVert_2$
Correlation	$1-\frac{(f(u)-\overline{f(u)})\cdot(f(v)-\overline{f(v)})}{\lVert(f(u)-\overline{f(u)})\rVert_2\lVert(f(v)-\overline{f(v)})\rVert_2}$
Chebyshev	$\max_i \lvert f_i(u)-f_i(v)\rvert$
Braycurtis	$\frac{\sum\lvert f_i(u)-f_i(v)\rvert}{\sum\lvert f_i(u)+f_i(v)\rvert}$
Manhattan	$\sum_i \lvert f_i(u)-f_i(v)\rvert$
Canberra	$\sum_i \frac{\lvert f_i(u)-f_i(v)\rvert}{\lvert f_i(u)\rvert+\lvert f_i(v)\rvert}$
Sqeuclidean	$\lVert f(u)-f(v)\rVert_2^2$

Table 14: Pairwise vector operations, $f_i(u)$ represents the i-th component of $f(u)$. Note that these operations can be applied to nodes' attributes and entropies summarized from posteriors as well.

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

Table 15: Prediction results for Attack-0 on all the 8 datasets with Correlation distance.

Dataset	Precision	Recall	F1-Score	AUC
AIDS	0.524	0.996	0.687	0.691
COX2	0.523	0.987	0.684	0.867
DHFR	0.555	0.977	0.708	0.765
ENZYMES	0.501	1.000	0.667	0.630
PROTEINS_full	0.540	0.998	0.701	0.815
Citeseer	0.788	0.991	0.878	0.959
Cora	0.777	0.966	0.861	0.929
Pubmed	0.691	0.965	0.806	0.874

Table 16: Average AUC with standard deviation for Attack-1 with different GCN structures on all the 8 datasets. Results with respect to the best performing shadow dataset are reported.

Dataset	Shadow Dataset	AUC
AIDS	PROTEINS_full	0.729 ± 0.013
COX2	Citeseer	0.760 ± 0.026
DHFR	COX2	0.792 ± 0.005
ENZYMES	AIDS	0.732 ± 0.009
PROTEINS_full	COX2	0.808 ± 0.034
Citeseer	Cora	0.924 ± 0.006
Cora	Citeseer	0.916 ± 0.002
Pubmed	Citeseer	0.840 ± 0.001

Table 17: Average Precision and Recall with standard deviation for Attack-1. Results with respect to the best performing shadow dataset are reported.

Dataset	Shadow Dataset	Precision	Recall
AIDS	ENZYMES	0.725 ± 0.044	0.505 ± 0.110
COX2	PROTEINS_full	0.828 ± 0.013	0.686 ± 0.100
DHFR	COX2	0.691 ± 0.015	0.704 ± 0.022
ENZYMES	AIDS	0.639 ± 0.023	0.615 ± 0.046
PROTEINS_full	Citeseer	0.750 ± 0.022	0.800 ± 0.055
Citeseer	Cora	0.871 ± 0.005	0.958 ± 0.005
Cora	Citeseer	0.854 ± 0.003	0.883 ± 0.008
Pubmed	Cora	0.765 ± 0.009	0.897 ± 0.012

Table 18: Average Precision and Recall with standard deviation for Attack-3.

Dataset	Precision	Recall
AIDS	0.874 ± 0.006	0.966 ± 0.005
COX2	0.846 ± 0.004	0.922 ± 0.005
DHFR	0.847 ± 0.007	0.877 ± 0.009
ENZYMES	0.761 ± 0.003	0.871 ± 0.004
PROTEINS_full	0.856 ± 0.006	0.943 ± 0.004
Citeseer	0.895 ± 0.003	0.946 ± 0.005
Cora	0.858 ± 0.002	0.917 ± 0.008
Pubmed	0.869 ± 0.008	0.892 ± 0.014

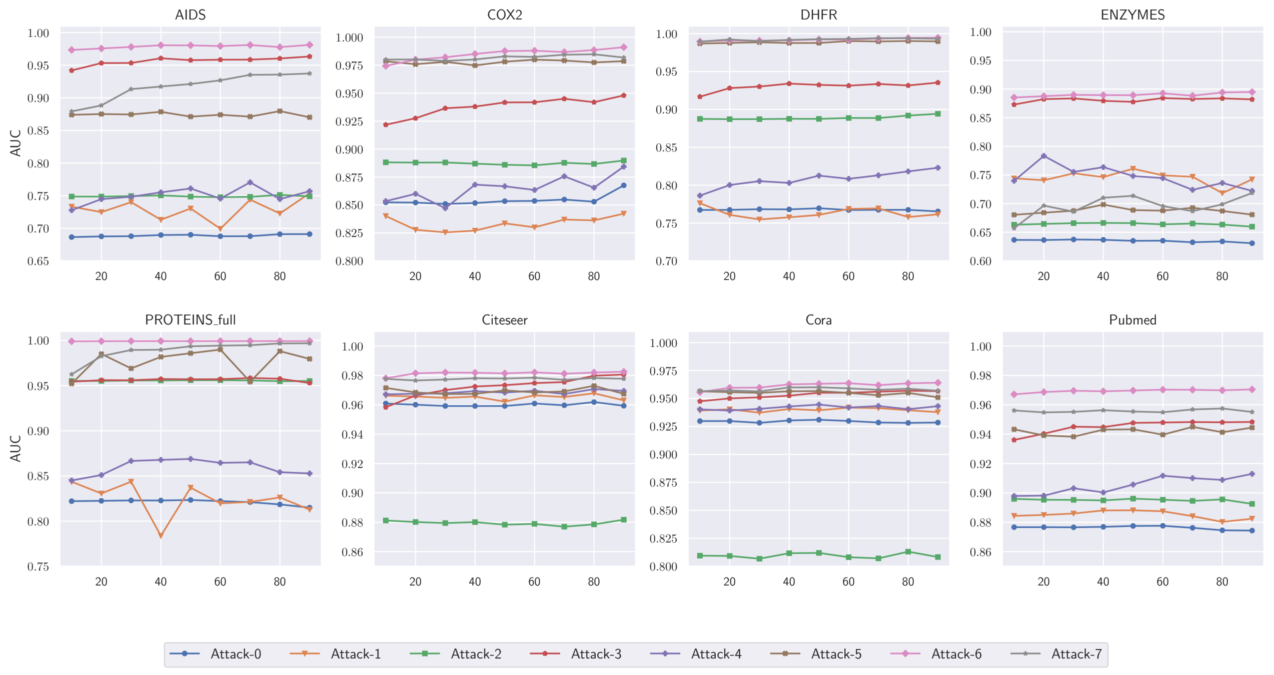


Figure 7: The relationship between the ratio of attack training dataset in the attack dataset and the attacks' AUC scores on all the 8 datasets. The x-axis represents the ratio and the y-axis represents the AUC score.

Table 19: Average Precision and Recall with standard deviation for Attack-4. Results with respect to the best performing shadow dataset are reported.

Dataset	Shadow Dataset	Precision	Recall
AIDS	DHFR	0.688 ± 0.013	0.628 ± 0.046
COX2	DHFR	0.787 ± 0.009	0.835 ± 0.033
DHFR	COX2	0.726 ± 0.008	0.793 ± 0.015
ENZYMES	AIDS	0.637 ± 0.025	0.683 ± 0.041
PROTEINS_full	Pubmed	0.686 ± 0.045	0.955 ± 0.020
Citeseer	Cora	0.874 ± 0.004	0.956 ± 0.004
Cora	Citeseer	0.854 ± 0.002	0.896 ± 0.004
Pubmed	Citeseer	0.790 ± 0.009	0.877 ± 0.012

Table 21: Average Precision and Recall with standard deviation for Attack-6.

Dataset	Precision	Recall
AIDS	0.907 ± 0.002	0.986 ± 0.002
COX2	0.935 ± 0.004	0.994 ± 0.001
DHFR	0.972 ± 0.001	0.995 ± 0.002
ENZYMES	0.770 ± 0.004	0.886 ± 0.009
PROTEINS_full	0.988 ± 0.002	0.998 ± 0.001
Citeseer	0.900 ± 0.008	0.933 ± 0.006
Cora	0.878 ± 0.003	0.930 ± 0.003
Pubmed	0.903 ± 0.004	0.920 ± 0.003

Table 20: Average Precision and Recall with standard deviation for Attack-5. Results with respect to the best performing shadow dataset are reported.

Dataset	Shadow Dataset	Precision	Recall
AIDS	PROTEINS_full	0.854 ± 0.003	0.663 ± 0.005
COX2	DHFR	0.941 ± 0.004	0.923 ± 0.022
DHFR	COX2	0.973 ± 0.004	0.942 ± 0.025
ENZYMES	Citeseer	0.608 ± 0.005	0.675 ± 0.013
PROTEINS_full	COX2	0.996 ± 0.003	0.061 ± 0.055
Citeseer	Cora	0.888 ± 0.006	0.885 ± 0.005
Cora	Citeseer	0.867 ± 0.006	0.892 ± 0.009
Pubmed	Cora	0.824 ± 0.010	0.913 ± 0.014

Table 22: Average Precision and Recall with standard deviation for Attack-7. Results with respect to the best performing shadow dataset are reported.

Dataset	Shadow Dataset	Precision	Recall
AIDS	COX2	0.870 ± 0.003	0.781 ± 0.013
COX2	DHFR	0.941 ± 0.004	0.966 ± 0.009
DHFR	COX2	0.972 ± 0.002	0.994 ± 0.005
ENZYMES	AIDS	0.617 ± 0.012	0.693 ± 0.036
PROTEINS_full	ENZYMES	0.955 ± 0.004	0.974 ± 0.010
Citeseer	Cora	0.898 ± 0.003	0.913 ± 0.008
Cora	Citeseer	0.874 ± 0.004	0.911 ± 0.005
Pubmed	Citeseer	0.881 ± 0.006	0.901 ± 0.010