Model Stealing Attacks Against Inductive Graph Neural Networks

Yun Shen[1]* Xinlei He[2]* Yufei Han[3] Yang Zhang[2]

[1]Norton Research Group   [2]CISPA Helmholtz Center for Information Security   [3]INRIA

Abstract—**Many real-world data come in the form of graphs. Graph neural networks (GNNs), a new family of machine learning (ML) models, have been proposed to fully leverage graph data to build powerful applications. In particular, the inductive GNNs, which can generalize to unseen data, become mainstream in this direction. Machine learning models have shown great potential in various tasks and have been deployed in many real-world scenarios. To train a good model, a large amount of data as well as computational resources are needed, leading to valuable intellectual property. Previous research has shown that ML models are prone to model stealing attacks, which aim to steal the functionality of the target models. However, most of them focus on the models trained with images and texts. On the other hand, little attention has been paid to models trained with graph data, i.e., GNNs. In this paper, we fill the gap by proposing the first model stealing attacks against inductive GNNs. We systematically define the threat model and propose six attacks based on the adversary's background knowledge and the responses of the target models. Our evaluation on six benchmark datasets shows that the proposed model stealing attacks against GNNs achieve promising performance.[1]**

I. INTRODUCTION

Many real-world data come in the form of graphs, such as molecular graphs [39] and social networks [60]. The graph-structured data contains nodes with features and edges that represent the relationship between them. To fully unleash the potential of graph data, a new family of machine learning (ML) models, namely graph neural networks (GNNs), has been proposed [23], [42], [81], [91], [92]. Compared to classical machine learning models, e.g., convolutional neural networks (CNNs) and recurrent neural networks (RNNs), which are designed to process images and texts, GNNs offer state-of-the-art performance by taking both node features as well as graph structures into consideration.

Prior work unveiled that machine learning models are vulnerable to *model stealing attacks* [34], [51], [52], [70], [74], where an adversary with query access to a target model can steal its parameters or functionality. Concretely, the adversary first crafts a number of queries as the input to the target model's API and obtains the corresponding outputs. Then, a local surrogate model is trained on the paired data (input, output). As such, the surrogate model may not only violate the intellectual property of the target model but also serve as a stepping stone for further attacks like membership inference [5], [8], [13], [29], [35], [45], [48], [58], [59], [63] and adversarial examples [6], [22], [54], [55], [69], [79].

Notably, most of the current efforts on model stealing attacks concentrate on ML models with images and text data [41], [51], [52], [70], [74]. On the other hand, the potential model stealing risks of GNNs have been largely understudied.

There exists some preliminary work on model stealing attacks against GNNs [16], [78]. They focus on *transductive* GNNs and assume that the attackers have access to the training process of the target model, in which the training and query graphs are used to train the target model. As such, GNN model stealing attacks in a transductive setting are unrealistic.

In this paper, we concentrate on a more realistic and popularly deployed GNN setting, i.e., *inductive* GNNs, which can generalize well to unseen nodes [25], [73], [85]. In this setting, the adversary only queries the target model via remotely accessible API. They do not tamper with the training process of the target model. Note that in this paper, we focus on node classification tasks.

When an adversary uses a *query graph* (i.e., a graph induced by a set of query nodes) to query the target model, they could face different types of *responses* from the target model, ranging from the posterior distribution over the possible labels of query nodes (called predicted posterior probability) to 2-dimensional t-SNE [72] projection (for graph visualization). Therefore, it is important to summarize a complete taxonomy of the threats for model stealing attacks against GNNs. Also, it is challenging to design a general attack methodology that can be applied to different attack scenarios. Moreover, in reality, the graph structural information of the query graph can be missing. It is inevitably harder for the adversary to launch attacks against GNNs.

To tackle these challenges, we make the following contributions in this paper. We first systematically define the threat model of model stealing attacks against inductive GNNs by categorizing the adversary's knowledge into two dimensions, i.e., query graph and target model's response (see Section III). Concretely, we assume that the adversary has a query graph that contains a number of nodes and their features. The node features come from the same distribution of the graph used to train the target model. However, the graph structure of the query graph may be missing (i.e., the edges connecting the nodes may not be available). Regarding the target model's response, we consider three cases, i.e., *the predicted posterior probability*, *the node embedding vector*, or the 2-dimensional *t-SNE projection*. In turn, we have six attack scenarios under our threat model.

---

[1]Our code is available at https://github.com/xinleihe/GNNStealing.

We propose two types of attacks, i.e., Type I and Type II, based on the information provided by the query graph (see Section IV-A). Each type has three variants depending on the target model's responses. We design a general attack framework that can be applied to all the scenarios. Concretely, the framework is assembled with two major components. The first component is used to learn the discrete graph structure if the structural information is not available in the query graph. Then, the second component builds a surrogate model by jointly learning from the nodes' features and the response of the target model.

Ideally, the surrogate model should achieve both high accuracy and high fidelity whereby accuracy measures the prediction correctness [54], [70] and fidelity measures the prediction agreement between the target model and the surrogate model [34], [41]. We evaluate all our attacks on three popular inductive GNN models including GraphSAGE [25], Graph Attention Network (GAT) [73], and Graph Isomorphism Network (GIN) [85] with six benchmark datasets, i.e., DBLP [53], Pubmed [61], Citeseer Full [21], Coauthor Physics [62], ACM [76], and Amazon Co-purchase Network for Photos [50]. Extensive experiments demonstrate that our model stealing attacks consistently achieve strong performance in both types of attacks. For instance, when the target model is GIN trained on the Pubmed dataset and the response is embeddings, our Type I attack achieves a 0.877 accuracy score and a 0.906 fidelity score (see Section V-B). In particular, when the aforementioned target model's response is the t-SNE projection, our attack still achieves strong performance with a 0.823 accuracy score and a 0.846 fidelity score. Moreover, we empirically demonstrate that even without graph structure information, i.e., Type II attacks, the adversary could still launch effective attacks, extracting high-accuracy and high-fidelity surrogate models (see Section V-C). This further demonstrates the severe model stealing risks of GNN models.

In summary, we make the following contributions.

- Our work is the first research effort to perform model stealing attacks against inductive GNNs.
- We systematically define the threat model to characterize an adversary's background knowledge along two dimensions. Moreover, we propose six different attack scenarios based on the adversary's different background knowledge.
- Extensive evaluation on three popular inductive GNN models and six benchmark graph datasets demonstrates the efficacy of our attacks.

II. BACKGROUND

A. Notations

We define a labelled, undirected, unweighted, attributed graph as $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathbf{C})$, where $\mathbf{V} = \{v_1, v_2, ..., v_n\}$ denotes the set of nodes, $\mathbf{E} \subseteq \{(v, u)|v, u \in \mathbf{V}\}$ denotes the set of edges, $\mathbf{x}_i \in \mathbf{X}$ denotes the feature of node v_i, and one-hot vectors $\mathbf{c}_i \in \mathbf{C}$ denotes the label of node v_i. We denote $\mathbf{A} \in \{0,1\}^{n \times n}$ as the adjacency matrix, where $\mathbf{A}_{vu} = 1, \forall(v, u) \in \mathbf{E}$. As such, $\mathbf{G}$ can also be represented as $\mathbf{G} = (\mathbf{A}, \mathbf{X}, \mathbf{C})$. The original graph used to train a GNN

TABLE I: Summary of the notations used in this paper. We use lowercase letters to denote scalars, bold lowercase letters to denote vectors and bold uppercase letters to denote matrices.

Notation	Description
$\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathbf{C})$	graph
$v, u \in V$	node
$n = |\mathbf{V}|$	number of nodes
$\mathbf{A} \in \{0,1\}^{n \times n}$	adjacency matrix
d	dimension of a node feature vector
b	dimension of a hidden node feature vector
$\mathbf{c}_i \in \mathbf{C}$	node label/class
$\mathcal{N}^l(v)$	l-hop neighborhood of v
$\mathbf{G}_v^l$	subgraph induced by l-hop neighborhood of v
$\mathbf{X} \in \mathbb{R}^{n \times d}$	node feature matrix
$\mathbf{x}_v \in \mathbb{R}^d$	feature vector of node v
$\mathbf{H} \in \mathbb{R}^{n \times b}$	hidden state matrix
$\mathbf{h}_v \in \mathbb{R}^b$	hidden state of v
$\mathbf{G}_O/\mathbf{G}_Q$	training/query graph
$\mathbf{R}$	query response
$\mathbf{\Upsilon} \in \mathbb{R}^{n \times 2}$	2-dimensional t-SNE projection
$\mathbf{\Theta} \in \mathbb{R}^{n \times |\mathbf{C}|}$	predicted posterior probability
$\mathcal{M}_T/\mathcal{M}_S$	target/surrogate GNN model

model is denoted as $\mathbf{G}_O$ (training graph). We use $\mathcal{N}^l(v)$ to denote l-hop neighborhood of v, and $\mathbf{G}_v^l$ to denote the subgraph induced by l-hop neighborhood of v. Besides, we denote the target GNN model as $\mathcal{M}_T$ and the surrogate GNN model as $\mathcal{M}_S$. The notations introduced here and in the following sections are summarized in Table I.

B. Preliminaries

Graph Neural Networks (GNNs). Representation learning of graph-structured data is challenging because both graph structure and node features carry important information. Graph Neural Networks (GNNs) provide an effective way to fuse information from network structure and node features. Most of the GNNs follow a neighborhood aggregation strategy, where the model iteratively updates the representation of a node through message passing and aggregating representations of its neighbors. After l iterations of aggregation, a node's representation, denoted as $\mathbf{h}_v$, captures the structural information within its l-hop network neighborhood. In practice, a GNN contains several graph convolutional layers. Each graph convolutional layer of a GNN model can be defined as follows:

$$\mathbf{h}_v^l = \text{AGGREGATE}(\mathbf{h}_v^{l-1}, \text{MSG}(\mathbf{h}_v^{l-1}, \mathbf{h}_u^{l-1})), u \in \mathcal{N}(v) \tag{1}$$

Note that the number of graph convolutional layers is equivalent to the l-hop network neighborhood that a GNN model can reach in the graph. Once trained, the GNN can map each node to an embedding vector. These node embeddings can be directly used for downstream machine learning tasks that can be categorized into three levels, i.e., node-level (e.g., node classification [25], [40], [73], [85]), link-level (e.g., link prediction [24], [56], [71]), and graph-level (e.g., graph classification [43], [87]).

Inductive GNN Models. There are two settings for training the GNNs, i.e., transductive setting and inductive setting. In

the transductive setting, a GNN learns from both labelled and unlabelled nodes in a single fixed graph at the training time and predicts the labels of those unlabelled nodes once the training is done, e.g., vanilla graph convolutional network (GCN) [40], DeepWalk [56], etc. However, transductive GNN models must be retrained if new nodes are introduced to the graph. A more popular one is the setting of inductive learning, where the learned GNN model can be generalized to the graphs that are previously unseen during the training procedure. The reusable GNN model avoids time-consuming retraining if a graph includes more nodes or even subgraphs. It facilitates the real-world practices of graph data analytics. We therefore focus on the inductive setting in our study. We briefly introduce three widely used inductive GNN models below.

- *GraphSAGE.* GraphSAGE proposed by Hamilton et al. [25] is the first inductive GNN model. Inspired by the Weisfeiler-Lehman test for graph isomorphism, Graph-SAGE generalizes the original GCN [40] into the inductive setting with different aggregation functions. Take widely used mean aggregation operator as an example, GraphSAGE can be defined as follows:

$$\mathbf{h}_v^l = \text{MEAN}(\mathbf{h}_v^{l-1} \cup \{\mathbf{h}_u^{l-1}, \forall u \in \mathcal{N}(v)\}) \quad (2)$$

- *Graph Attention Network (GAT).* It is straightforward to observe that GraphSAGE assigns the same weight to all neighbors (i.e., $1/\mathcal{N}(v)$) when aggregating v's neighborhood information. However, in practice, different nodes may play different roles in the target node embedding. Inspired by the attention mechanism in deep learning [1], Velickovic et al. [73] propose GAT that leverages multi-head attention to learn different attention weights and pays more attention to the important neighborhoods. Its aggregation function can be formulated as:

$$\mathbf{h}_v^l = \Big\|_{z=1}^{Z} \big(\sum_{u \in \mathcal{N}(v)} \alpha_{uv}^z \cdot \mathbf{W}^z \cdot \mathbf{h}_u^{l-1} \big) \quad (3)$$

where $\|$ is the concatenation operation, Z is the total number of projection heads in the attention mechanism, W^z is the linear transformation weight matrix, and α_{uv}^z is the attention coefficient calculated by the z-th projection head.

- *Graph Isomorphism Network (GIN).* GraphSAGE can be treated as an instance of the Weisfeiler-Lehman test. Xu et al. [85] propose Graph Isomorphism Network (GIN) to extend GraphSAGE with arbitrary aggregation functions on multi-sets. GIN is theoretically proven to be as powerful as the Weisfeiler-Lehman test of graph isomorphism. Its aggregation function can be represented as:

$$\mathbf{h}_v^l = (1 + \epsilon^{l-1}) \cdot \mathbf{h}_v^{l-1} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{l-1} \quad (4)$$

where ϵ is a learnable parameter to adjust the weight of node v.

The inductive GNNs usually employ shared weight parameters and neighborhood sampling to speed up the computation.

Responses by Inductive GNNs. An inductive GNN model learns the parameters of aggregation functions in different layers from the training data. Once trained, the learned GNN model can infer previously unseen data. Such capability paves the way for remotely deployed GNN models in the wild (e.g., GROVER [57], DGL [75]) to make inferences on graphs for customers via publicly accessible API. Moreover, a trained GNN model is often used to perform node embedding tasks, and the resulted node embeddings can then help to perform other downstream ML tasks (e.g., fine-tuning pretrained GNNs [30], model partitioning [64]) or graph visualization. Therefore, in this paper, we consider three query responses from a target GNN model when facing a query node, namely predicted posterior probability, embedding vector, and 2-dimensional t-SNE projections from the embedding [23], [42], [81], [91], [92]. Specifically, given a query node $v \in \mathbf{V}_Q$, we feed its l-hop subgraph (i.e., G_v^l) to a remote GNN model and obtain one of the three corresponding responses. All the query nodes with the edges between them can form a query graph/dataset, denoted as $\mathbf{G}_Q$. Note that $\mathbf{G}_Q$ is not necessarily connected and for each query node, the l-hop subgraph is extracted from $\mathbf{G}_Q$ only.

III. THREAT MODEL

In this section, we outline the threat model to characterize the adversary's background knowledge and the goal of the model stealing attack.

A. Attack Setting

We frame our attack in a *black-box* setting, which is the most challenging scenario for the adversary mentioned in previous work [27], [28], [34], [51], [63]. That is, the adversary has no knowledge of the target GNN model (e.g., model parameters, model architecture) and cannot tamper with its training process (e.g., training graph $\mathbf{G}_O$). Our attack setting is fundamentally different from the previous attacks [16], [78]. These attacks assume that the adversary can gain access to the target model's training process. However, such a strong assumption is unrealistic in the real world as it is impractical to expect an adversary can interfere with the target model at its training time. Note that in this paper we focus on the aforementioned three node-level query responses. The target model is an inductive GNN model which accepts node v's l-hop subgraph $\mathbf{G}_v^l$ as input and returns the corresponding response for the given node v, i.e., its predicted posterior probability, node embedding vector, or 2-dimensional t-SNE projection.

B. Adversary's Goal

Following the taxonomy defined by Jagielski et al. [34], the adversary's goal falls into two categories, i.e., theft and reconnaissance.

- The goal of the theft adversary is to build a surrogate model $\mathcal{M}_S$ that matches the accuracy of the target model $\mathcal{M}_T$ on the target task [54], [70]. The theft adversary's motivation is compromising the intellectual property and violating the confidentiality of the target model $\mathcal{M}_T$.

- Subtly different from the theft adversary, the reconnaissance adversary aims to build a surrogate model that closely matches the behavior of the target model. That is, $\mathcal{M}_S$ seeks an agreement to $\mathcal{M}_T$ on any input. A surrogate model $\mathcal{M}_S$ with high fidelity to $\mathcal{M}_T$ enables the adversary to leverage it as a stepping stone to launch further attacks. For instance, the adversary can craft adversarial examples using this $\mathcal{M}_S$ instead of risking potentially detectable queries to the target model $\mathcal{M}_T$ [54].

Note that the reconnaissance adversary's motivation is faithfully copying the behavior of $\mathcal{M}_T$ (e.g., $\mathcal{M}_S$ and $\mathcal{M}_T$ may make the same wrong/correct prediction of an input) though both adversaries intend to get close to the performance (i.e., accuracy) of the target model. We refer the audience to the work by Jagielski et al. [34] for additional discussions.

C. Adversary's Capability

We first assume an adversary can make queries to a target model $\mathcal{M}_T$. Given a query graph $\mathbf{G}_Q$, the adversary can query all its nodes with their corresponding l-hop subgraphs, then obtain the query responses $\mathbf{R}$. This assumption is in line with the adversarial machine learning setting, whereas the attackers exploit the remote target model $\mathcal{M}_T$ via publicly accessible API [27], [28], [52], [70]. The response $\mathbf{R}$ of the whole query graph $\mathbf{G}_Q$, depending on the target model's specification, may be returned in the form of a node embedding matrix (denoted as $\mathbf{H}$), a predicted posterior probability matrix (denoted as $\mathbf{\Theta}$), or a t-SNE projection matrix of $\mathbf{H}$ (denoted as $\mathbf{\Upsilon}$), where each row is a 2-dimensional vector. These three responses are representative of the real-world API output. For example, t-SNE and node embeddings are widely returned in the scenarios of graph visualization [32], transfer learning [94], federated learning [26], fine-tuning pretrained GNNs [30], and model partitioning where the target model is split into local and cloud parts bridged by embeddings information [64]. Moreover, these responses characterize three different levels of knowledge the adversary may gain access to in practice. For instance, the t-SNE projection matrix $\mathbf{\Upsilon}$ is usually used for data visualization. It contains the least information that an adversary can harvest. We later show that the adversary can still launch the attack through such data visualization function of a remote model.

Second, we assume that the query graph $\mathbf{G}_Q$ (both node features $\mathbf{X}_Q$ and graph structure $\mathbf{A}_Q$) is from the same distribution of the training graph $\mathbf{G}_O$ used to train the target model $\mathcal{M}_T$. Here we consider the term "same distribution" as $\mathbf{G}_O$ and $\mathbf{G}_Q$ are drowned randomly from the same dataset. We do not necessarily require $\mathbf{G}_O$ and $\mathbf{G}_Q$ to have the same graph characteristics. Besides, the nodes in the query graph $\mathbf{G}_Q$ does not need to be in the training graph $\mathbf{G}_O$. For example, both $\mathbf{G}_O$ and $\mathbf{G}_Q$ can be subgraphs sampled from social networks like Twitter but there is no overlap with each other. This assumption is in line with recent attacks to neural networks where an adversary uses part of a public dataset to exploit the target model [27], [34] and graphs are available in some domains (e.g., social networks and molecular graphs).

We further relax this assumption and demonstrate that the adversary can still launch effective attacks even without the graph structural information of $\mathbf{G}_Q$. This weak assumption of adversary knowledge makes our attack more practical in real-world scenarios. For instance, the adversary may compromise a user database and acquire their profile information. However, the relationships among the users, i.e., the graph structure, may not be revealed. We later demonstrate that the adversary can still launch high-performance model stealing attacks in this scenario (see Section V).

Notes. Our threat model is different from the *causative* and *evasion* adversarial attacks to GNNs [10], [14], [37], [66], [82], [83]. Those attacks allow the attackers to manipulate the training graphs in order to change the parameters of the target model, or modify the node features and/or graph structure to fool the inductive GNN models. Our attack is an instance of *exploratory attacks*. We do not tamper with the original training process. The goal is thereof not to change the parameters or fool the target GNN model. Instead, it is designed to steal a copy of a target inductive GNN model.

Our attack also differs from the existing model stealing attacks against *transductive GNNs* [16], [78] in several key aspects. First, these attacks [16], [78] focus on attacking transductive GNN models. Both methods assume that the query graph is part of the graph used for training the target GNN model and must be involved in the training process, hence unrealistic. Our attacks instead focus on a more realistic stealing attack scenario whereas the adversary only queries the target model via remotely accessible API. We do not tamper with the training process of the target model. In turn, our threat model is practical and fills the gap to understand if both theft and reconnaissance adversaries can steal inductive GNNs with high accuracy and high fidelity. Moreover, the existing methods [16], [78] are limited to the GCN model (i.e., model dependent) and rely on node predicted posterior probability scores to launch attacks. In contrast, our attack is model agnostic and can still successfully copy the target model's behavior with marginal information. Besides, our attack setting is different from Attack-3 proposed by Wu et al. [78]. Specifically, Attack-3 in [78] trains the surrogate model without querying the target model, while our attacks do interact with the target model. Note that it is impractical to attain the reconnaissance goal without interacting with the target model [28], [45]. Also, the target model considered by Wu et al. [78] is a transductive GNN while ours focuses on the inductive GNN (the difference between transductive and inductive GNN is described in Section II).

IV. Model Stealing Attack

In this section, we first outline 6 attack scenarios that can be launched by the adversary given different levels of knowledge. Then we propose our attack framework.

A. Attack Taxonomy

As outlined in Section III, the adversary has two main pieces of information at their disposal to launch the model
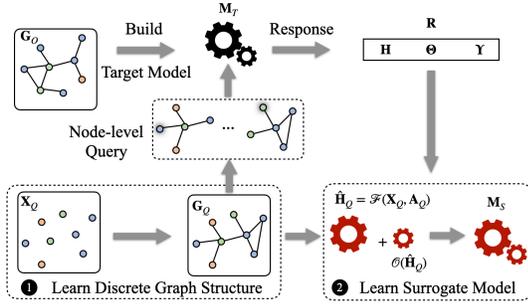
Fig. 1: Overview of model stealing attack against inductive GNNs.

stealing attack: the query graph $\mathbf{G}_Q = (\mathbf{A}_Q, \mathbf{X}_Q, \mathbf{C}_Q)$ and its corresponding query response $\mathbf{R}$ (i.e., node embedding matrix $\mathbf{H}$, predicted posterior probability matrix $\mathbf{\Theta}$, or t-SNE projection matrix $\mathbf{\Upsilon}$). Recall that the adversary may not have graph structural information of the query graph $\mathbf{G}_Q$ (i.e., the adjacency matrix $\mathbf{A}_Q$ may be missing). We thus have 6 possible attack scenarios falling into two types (see Table II).

- **Type I Attack**. When the adversary obtains the query graphs which are of the same distribution as the training graphs, they can launch a Type I attack against the target model. The application scenarios of Type I attack include side-effects prediction due to drug interactions [95], financial fraud detection [47], recommendation systems [80], etc. In these cases, the adversary can obtain the query graphs (e.g., drug-protein interaction graphs, transaction graphs, user-item purchase graphs) that are of the same distribution of the training graphs used to train the target models due to the wide availability of such graphs.

- **Type II Attack.** When the graph structural information is missing, the adversary can resort to a Type II attack to steal the target model. For instance, social networks such as Instagram or Tinder do not reveal the social relationship of (private) user accounts. However, an adversary can crawl users' profile information without social relations (i.e., graph structural information). Then they leverage Type II attacks to rebuild a relationship graph and steal the target models offered by these service providers. Besides, Type II attacks can also be used in stealthy insider threat scenarios. For instance, a company may enforce data segregation due to data privacy and security concerns (e.g., one department has user information and another has the relationship graph). To train a joint model, the company needs to perform vertical federated learning [86]. The insider attackers may leverage the information they can access (e.g., user information) to steal the joint model using our Type II attack.

It is straightforward to observe in Table II that the attacks are increasingly harder in the row order. For instance, the Type II.3 attack is the most challenging scenario from the

TABLE II: Attack Taxonomy. The attacks are increasingly harder in row order.

Attack	$\mathbf{A}_Q$	Response $\mathbf{R}$		
		$\mathbf{H}$	$\mathbf{\Theta}$	$\mathbf{\Upsilon}$
I.1	✓	✓		
I.2	✓		✓	
I.3	✓			✓
II.1	✗	✓		
II.2	✗		✓	
II.3	✗			✓

adversary's perspective since they only have a set of node features to start with. As such, they first need to restore the relationships among the nodes and build the query graph $\mathbf{G}_Q$, then leverage the t-SNE projection matrix obtained from $\mathcal{M}_T$ to steal the target model. Besides, model stealing attacks against the classical ML models focus on the scenario that the remote models return the predicted posterior probability. We note that the node embedding and the t-SNE projection-based query responses are the new attack surface of the inductive GNN models and outline the technical details in Section IV-B.

B. Attack Framework

To tackle the aforementioned challenges, we propose a unified attack framework as illustrated in Figure 1 to launch both Type I and II attacks. It has two components. The first component (❶ in Figure 1) learns the missing graph structure $\mathbf{A}_Q$. The output of the first component is a learned query graph $\mathbf{G}_Q$. This component is designed to facilitate Type II attacks, hence not required for Type I attacks. It is important to note that this reconstruction process is done locally at the attacker's side. They do not interact with the target model. The second component (❷ in Figure 1) learns a surrogate model from the response of the target model given the query graph $\mathbf{G}_Q$. The output of the second component is a learned surrogate model $\mathcal{M}_S$. We outline their details below.

1) Learn Discrete Graph Structure (❶)

Recall that the adversary does not have the adjacency matrix $\mathbf{A}_Q$ for the query graph $\mathbf{G}_Q$ at their disposal in Type II attacks (see Section III). However, the graph structure is necessary for the adversary to conduct the attack. As such, without a graph structure to piece $\mathbf{X}_Q$ together, the adversary must first build the adjacency matrix $\mathbf{A}_Q$ from query graph $\mathbf{G}_Q$ and then query the target model $\mathcal{M}_T$. The goal of this component is thereof to learn a high-quality discrete graph structure that enables the adversary to query and gain useful knowledge from the query response returned by the target model.

One common approach is to create a k-nearest neighbor (kNN) graph from node features $\mathbf{X}_Q$. However, the efficacy of the resulting kNN graph and consequent GNN model rests on the choice of k (e.g., kNN graphs often produce nodes with extremely high degrees) and the chosen similarity measure over the node features. To this end, we leverage the IDGL framework proposed by Chen et al. [11] to learn a query graph $\mathbf{G}_Q$ by minimizing a joint loss function combining both the task-dependent prediction loss and the graph regularization

loss. Note that task-dependent prediction loss is flexible, and can be tailored to use different loss functions (e.g., node classification loss or link prediction loss). The graph regularization loss controls the smoothness, connectivity, and sparsity of the resulting graph. Additional details can be found in [11].

To launch Type II attacks, the adversary first initiates a kNN graph constructed based on multi-head weighted cosine similarity, then utilizes the IDGL framework to search for a hidden graph structure that augments the initial kNN graph structure using the aforementioned joint loss function. Note that the kNN graph is only used to seed the initial graph structure. The final learned graph structure is optimized during the learning process and may not contain the edges from the original kNN graph. This component's output is a learned query graph $\mathbf{G}_Q$.

2) Learn Surrogate Model (❷)

The goal of the second component is to learn a surrogate GNN model from the query response returned by the target model given a query graph $\mathbf{G}_Q$. To this end, we first discuss our observation of three state-of-the-art inductive GNN models, namely *GraphSAGE*, *GIN*, and *GAT*. We then propose a unified framework to learn surrogate models given both Type I and Type II attacks.

Observation. According to the convolution operations defined on graphs, graph neural networks can be categorized as spectral approaches and spatial approaches [12], [81], [92], [93]. For spectral approaches, the graph is represented with a Laplacian matrix according to the spectral theories with the convolution operation defined in the sequence domain via Fourier transform. Different from spectral approaches, spatial approaches define graph convolutions by collective information propagation (i.e., propagating node information along edges) and perform convolution by considering node neighborhoods. Leveraging the insights from [12], [81], [92], [93], in spatial-based GNN methods, at l-th layer, node embedding $\mathbf{h}_v^l$ is iteratively updated using Equation 5 where Φ and Ψ are weight functions, and η is a normalization factor.

$$\mathbf{h}_v^l = \Phi(v)\mathbf{h}_v^{l-1} + \sum_{u \in \mathcal{N}(v)} \Psi(u)\frac{\mathbf{h}_u^{l-1}}{\eta} \quad (5)$$

If Ψ assigns the same weight to all neighbors Equation 5 can be rewritten in matrix form as Equation 6, where $\mathbf{I}$ is an identity matrix and $\tilde{\mathbf{A}}$ is a normalized form of adjacency matrix $\mathbf{A}$. Note that $\mathbf{H}^0 = \mathbf{X}$.

$$\mathbf{H}^l = (\Phi\mathbf{I} + \Psi\tilde{\mathbf{A}})\mathbf{H}^{l-1} \quad (6)$$

It is straightforward to prove that GraphSAGE (Equation 2 in Section II) can be written in matrix form as Equation 7, where $\Phi = 1$, $\Psi = 1$, and $\tilde{\mathbf{A}}$ is a random-walk normalized Laplacian matrix.

$$\mathbf{H}^l = (\mathbf{I} + \tilde{\mathbf{A}})\mathbf{H}^{l-1} \quad (7)$$

Similarly, GIN (Equation 4 in Section II) can be written as Equation 8, where $\Phi = 1 + \epsilon$, $\Psi = 1$, $\mathbf{A}$ is unnormalized adjacency matrix.

$$\mathbf{H}^l = ((1+\epsilon)\mathbf{I} + \mathbf{A})\mathbf{H}^{l-1} \quad (8)$$

In the same way, GAT (Equation 3 in Section II) can be written as Equation 9, where $\Psi = \mathbf{\Xi}$ is a learnable neighbor weight matrix and $\Phi = 0$.

$$\mathbf{H}^l = (\mathbf{\Xi} \otimes \mathbf{A})\mathbf{H}^{l-1} \quad (9)$$

Given Equation 7, Equation 8, and Equation 9, we can see that there exists an intrinsic connection among GraphSAGE, GAT, and GIN (i.e., they are special cases of Equation 6). It is evidential that these models are spatial-based GNN models through adopting different designs for feature aggregation. This observation enables us to design a unified attack framework to leverage the query graph $\mathbf{G}_Q$ as the data and the query response $\mathbf{R}$ as the supervised information to build the surrogate model in a unified manner.

Learning Framework. Our attack framework is illustrated in Figure 1. The surrogate model $\mathcal{M}_s$ in ❷ consists of two modules. The first module is a customized inductive GNN model (denoted as $\mathcal{F}$) taking all nodes' l-hop subgraphs from $\mathbf{G}_Q$ as the training data and the query response $\mathbf{R}$ as the supervised information. Since the responses from $\mathcal{M}_T$ are vectors in Euclidean space, they reflect the spatial connectivity among nodes either from the graph connectivity perspective (t-SNE projection or node embedding), or from the node label perspective (the predicted posterior probability). That is, the nodes that are close or connected in the query graph $\mathbf{G}_Q$ should be close in $\mathbf{\Upsilon}$ or $\mathbf{H}$, and the nodes that are of the same label should be close in $\mathbf{\Theta}$.

Following the above observation, the key idea of our attack is that we ignore the response types and uniformly treat all three possible responses as embedding vectors. As such, for the first module, the goal is to minimize the RMSE loss ($\mathcal{L}_R$) between $\hat{\mathbf{H}}_Q$ and $\mathbf{R}$ as shown in Equation 10.

$$\begin{aligned} \hat{\mathbf{H}}_Q &= \mathcal{F}(\mathbf{X}_Q, \mathbf{A}_Q) \\ \mathcal{L}_R &= \frac{1}{n_Q}\|\hat{\mathbf{H}}_Q - \mathbf{R}\|_{2,1} \end{aligned} \quad (10)$$

where n_Q denotes the number of nodes of the query graph G_Q. Here $\hat{\mathbf{H}}$ keeps the same dimension as $\mathbf{R}$. The rationale of using the RMSE loss is that $\mathcal{F}$ is optimized to maintain the similar spatial connectivity among the nodes in $\hat{\mathbf{H}}_Q$ as suggested by $\mathbf{R}$. Note that the output from the first component cannot be directly used for node classification tasks. In light of this, we employ an MLP as the classifier (denoted as $\mathcal{O}$). It takes the output from the first module (i.e., $\hat{\mathbf{H}}_Q$) as the input and $\mathbf{C}_Q$ as the supervision information to minimize the prediction error ($\mathcal{L}_P$) as shown in Equation 11.

$$\mathcal{L}_P = -\frac{1}{n_Q}\sum_{v \in \mathbf{G}_Q}\sum_{i \in |\mathbf{C}_Q|} c_i log[\mathcal{O}(\mathbf{h}_v)_i] \quad (11)$$

TABLE III: Summary of datasets.

Dataset	$|\mathbf{V}|$	$|\mathbf{E}|$	$|\mathbf{X}|$	$|\mathbf{C}|$	Density
DBLP	17,716	105,734	1,639	4	0.0007
Pubmed	19,717	88,648	500	3	0.0005
Citeseer	4,230	5,358	602	6	0.0006
Coauthor	34,493	495,924	8,415	5	0.0008
ACM	3,025	26,256	1,870	3	0.0057
Amazon	7,650	143,663	745	8	0.0049

We optimize the first module with $\mathcal{L}_R$, then we freeze it and optimize the second module with $\mathcal{L}_P$. The two modules are then chained together as our surrogate model. Followed by Orekondy et al. [52], we assume that $\mathbf{C}_Q$ is obtained by the adversary.

Our attack design enjoys two-fold flexibility. *First*, our attack allows the adversary to conduct the attack without knowing the target model's architecture. This makes the threat model closer to the real-world scenario where the adversary only has query access to the target model. *Second*, our attack framework shows that node embedding and 2-dimensional t-SNE projection as the query response can be the new attack surface of the inductive GNN models. Notably, t-SNE projection is for data visualization purposes. It barely unveils the graph structural information to the adversary. However, we demonstrate that our attack can still successfully copy the target model's behavior from such marginal information.

V. EVALUATION

In this section, we perform an in-depth analysis of the proposed model stealing attack against inductive GNNs. We first introduce the experimental setup, and present the evaluation results for Type I and Type II attacks from both theft and reconnaissance adversary's perspectives. We then explore how various query budgets may affect the attack performance. Finally, we study how different hyperparameters of the surrogate model may influence the attack performance.

A. Experimental Setup

Datasets. We use 6 public datasets to evaluate the performance of our attack, including DBLP [53], Pubmed [61], Citeseer Full (abbreviated as Citeseer) [21], Coauthor Physics (abbreviated as Coauthor) [62], ACM [76], and Amazon Co-purchase Network for Photos (abbreviated as Amazon) [50]. These datasets are widely employed as benchmark datasets to evaluate the performance of GNNs [25], [40], [85]. Among them, DBLP, Pubmed, and Citeseer are citation networks with nodes representing publications and edges indicating citations among these publications. Coauthor is a user interaction network where nodes represent the users and edges represent interactions between them. ACM and Amazon are cooperative networks where nodes represent the papers/items and there is an edge between two nodes if they have the same author or purchased together. We use these datasets to verify the efficacy of our attacks given different graph characteristics (e.g., graph size, node feature size, number of classes, etc.). Statistics of these datasets are summarized in Table III.

Dataset Configuration. For each dataset, we split them into three parts. The first part consists of 20% randomly sampled nodes that are used to train the target model $\mathcal{M}_T$. The second part consists of 30% randomly sampled nodes, forming our query graph $\mathbf{G}_Q$. We further show that our attacks are still effective with fewer nodes to form the query graph (see Figure 6). The third part consists of the rest 50% of the nodes, functioning as the testing data for both $\mathcal{M}_T$ and $\mathcal{M}_S$. This setting matches the inductive learning on evolving graphs as laid out in [25].

Target Model ($\mathcal{M}_T$). We use GIN, GAT, and GraphSAGE as our target models' architectures in our evaluation. For reproducibility purposes, we outline the details below.

- **GIN.** We use a 3-layer GIN model with a fixed neighborhood sample size of 10 at each layer. For the first hidden layer, we set the hidden unit size to 256. For the second layer, we set the hidden unit size to the embedding size (i.e., 64, 128, or 256 in our experimental setting). The final layer is used for classification.
- **GAT.** We use a 3-layer GAT model with a fixed neighborhood sample size of 10 at each layer. The first layer consist of 4 attention heads and the hidden unit size is 256. The second layer consists of 4 attention heads and the hidden unit size is the embedding size. The final layer is used for classification following the original design [73].
- **GraphSAGE.** Following Hamilton et al. [25], we use a 2-layer GraphSAGE with neighborhood sample sizes of 25 and 10 respectively. For the first hidden layer, we set the hidden unit size to 256. For the second layer, we set the hidden unit size to the embedding size. Each layer employs a GCN aggregator and uses 0.5 dropout rate to prevent overfitting. Finally, we use a linear transformation layer for classification.

All models use cross-entropy as the loss function, ReLU as the activation function between layers, and Adam as the optimizer with an initial learning rate of 0.001. We train all models for 200 epochs and select the best models with the highest validation accuracy. All models above follow the design specifications outlined in the respective papers.

Query Response ($\mathbf{R}$). For the node embedding $\mathbf{H}$, we fix the sizes to three commonly used values, i.e., 64, 128, and 256. For the node predicted posterior probability $\boldsymbol{\Theta}$, the dimension sizes (i.e., the number of classes) are dataset dependent and outlined in Table III. For the node projection $\boldsymbol{\Upsilon}$, we use t-SNE to project the node embedding $\mathbf{H}$ into 2-dimensional vectors, which cover most of the data visualization cases.

Surrogate Model ($\mathcal{M}_S$). Recall our attack design in Section IV-A, the surrogate model consists of two components to provide extensibility. The first component is a customized GNN model taking the subgraph extracted from $\mathbf{G}_Q$ as the input and using the RMSE as its loss function. The second component is a 2-layer MLP with the hidden unit size of 100. It takes the output from the first component as the input and uses the cross-entropy as its loss function. Both

components use Adam optimizer with a learning rate of 0.001. We train the first and the second components for 200 epochs and 300 epochs, respectively. For evaluation purposes, we use customized GraphSAGE, GAT, and GIN models as the first component in our surrogate model. The details are outlined below.

- **GIN.** We use a 2-layer GIN model with neighborhood sample sizes of 10 and 50 respectively. The hidden unit size is 256 for the first layer. For the second layer, we set the hidden unit size to the size of the query response.
- **GAT.** We use a 2-layer GAT model with neighborhood sample sizes of 10 and 50 respectively. Both the first and the second layers consist of 4 attention heads and we follow the same hidden unit size as GIN we mentioned above.
- **GraphSAGE.** We use a 2-layer GraphSAGE with neighborhood sample sizes of 10 and 50 respectively and follow the same hidden unit size as GIN we mentioned above.

Query Graph Reconstruction Configuration. We use the IDGL framework proposed by Chen et al. [11] to learn the missing graph structural information, i.e., $\mathbf{A}_Q$, of the query graph $\mathbf{G}_Q$ for Type II attacks. For simplicity, we use node classification loss and leave experimenting with other loss functions as future work. The initial k of kNN graph is set to 24. We use a 2-layer (hidden unit size is set to 256) GraphSAGE with GCN aggregator to learn the node embedding. Weighted cosine similarity matrices with 8 attention heads are employed to decide if there exists an edge between two node embeddings during the graph learning process. The graph structure and the GraphSAGE parameters are jointly and iteratively learned by minimizing a hybrid loss function combining both the node classification loss and the graph regularization loss. We set the cutoff value ϵ to 0.99 to identify the final edges in the learned adjacency matrix $\mathbf{A_Q}$ of the query graph $\mathbf{G}_Q$.

Metrics. Following the taxonomy defined by Jagielski et al. [34], we use two metrics, i.e., accuracy and fidelity, to evaluate the performance of our attack. We use accuracy (i.e., the number of correct predictions made divided by the total number of predictions made) as our evaluation metric of theft adversary. Accuracy has been dominantly used in evaluating node classification performance of GNNs [25], [40], [73]. Recall that the goal of reconnaissance adversary is to closely match the behavior of the target model (see Section III), we use fidelity (i.e., the number of predictions agreed by both $\mathcal{M}_S$ and $\mathcal{M}_T$) as the second evaluation metric of our attack [34], [38]. Both metrics are normalized between 0 and 1. Higher scores imply better performance.

Runtime Configuration. Note that we have 27 different combinations for each response in each dataset (i.e., combinations of three target models, three responses, and three surrogate models). All the experiments in this paper are repeated 5 times. For each run, we follow the same data configuration and report

TABLE IV: The performance of the original classification tasks on all the 6 datasets using 3 different GNN structures.

Dataset	$\mathcal{M}_T$		
	GIN	GAT	SAGE
DBLP	0.872	0.838	0.858
Pubmed	0.924	0.905	0.909
Citeseer	0.910	0.910	0.918
Coauthor	0.953	0.965	0.956
ACM	0.929	0.935	0.937
Amazon	0.856	0.953	0.937

the mean as well as the standard deviation of the aforementioned two metrics to evaluate the attack performance.

B. Performance Evaluation: Type I Attacks

We first summarize the accuracy of the target models for the original node classification tasks in Table IV. We can observe that all GNN models achieve good performance on all datasets, which demonstrates that jointly considering node features and graph structure are effective for classification. We then show the accuracy and fidelity of Type I attacks in Table V. Due to space limitations, we only show the attack results when the adversary uses GraphSAGE as the surrogate model. The performance results using GIN and GAT as the surrogate models follow similar patterns and can be found in Appendix Section A.

Accuracy. As we can see from Table V, Type I attacks can build surrogate models close to the target models given the response is predicted posterior probability or node embeddings (i.e., Type I.1 and Type I.2 attacks respectively). Take the Pubmed dataset as an example, the target models (i.e., GIN, GAT, and GraphSAGE) respectively achieve 0.924, 0.905, and 0.909 accuracy scores (see Table IV), while the surrogate models can consistently achieve at least 0.869 accuracy score (see Table V). This represents an approximately 0.04 accuracy score drop in all cases compared to the target models. We can also observe that Type I attacks can build surrogate models that offer usable accuracy even the response is a 2-dimensional t-SNE projection matrix (i.e., Type I.3 attack). For instance, on the Pubmed dataset, the surrogate models achieve 0.823, 0.743, and 0.844 accuracy scores respectively. This represents a 0.162 accuracy score drop in the worst case when the target model is GAT. For the rest of the five datasets, we also observe a subtle performance drop in Type I.3 attacks. Also, such performance drop, compared to Type I.1 and Type I.2 attacks, is expected since each t-SNE projection is only a 2-dimensional vector, which leads to additional information loss. Overall, our results show that all three Type I attacks can build usable surrogate models.

Given our experimental configuration (i.e., three target models and three surrogate models), there are 9 different combinations for each response type given a single dataset. Such configuration enables us to understand if the adversary can reliably steal target models in different circumstances. Using the Pubmed dataset as an example, we summarize the attack accuracy performance in Figure 2. We can see that in general, the adversary can build accurate surrogate

TABLE V: The accuracy and fidelity scores of Type I attacks using different response information on all the 6 datasets. Both average values and standard deviations are reported. The accuracy differences (in parenthesis) of the surrogate models to the target models are also reported. We use GraphSAGE as the surrogate model.

Dataset	$\mathcal{M}_S$ (SAGE)	$\mathcal{M}_T$					
		GIN		GAT		SAGE	
		Accuracy	Fidelity	Accuracy	Fidelity	Accuracy	Fidelity
DBLP	Projection	0.704±0.030 (-0.168)	0.727±0.032	0.682±0.003 (-0.156)	0.690±0.002	0.708±0.018 (-0.150)	0.748±0.018
	Prediction	0.769±0.006 (-0.103)	0.799±0.006	0.787±0.006 (-0.051)	0.827±0.005	0.810±0.005 (-0.048)	0.884±0.004
	Embedding	0.761±0.003 (-0.111)	0.790±0.003	0.793±0.006 (-0.045)	0.835±0.006	0.827±0.003 (-0.031)	0.904±0.003
Pubmed	Projection	0.823±0.035 (-0.101)	0.846±0.035	0.743±0.046 (-0.162)	0.733±0.045	0.844±0.028 (-0.065)	0.888±0.034
	Prediction	0.875±0.004 (-0.049)	0.903±0.001	0.869±0.002 (-0.036)	0.898±0.003	0.871±0.002 (-0.038)	0.924±0.003
	Embedding	0.877±0.003 (-0.047)	0.906±0.004	0.875±0.004 (-0.030)	0.905±0.004	0.881±0.002 (-0.028)	0.941±0.003
Citeseer	Projection	0.685±0.022 (-0.225)	0.668±0.024	0.691±0.014 (-0.220)	0.667±0.012	0.700±0.013 (-0.219)	0.707±0.010
	Prediction	0.802±0.008 (-0.108)	0.806±0.011	0.866±0.003 (-0.045)	0.887±0.004	0.878±0.006 (-0.041)	0.909±0.006
	Embedding	0.804±0.008 (-0.106)	0.811±0.008	0.877±0.004 (-0.034)	0.898±0.005	0.883±0.006 (-0.036)	0.914±0.006
Coauthor	Projection	0.845±0.041 (-0.108)	0.849±0.041	0.816±0.037 (-0.149)	0.817±0.038	0.832±0.031 (-0.124)	0.844±0.031
	Prediction	0.942±0.002 (-0.011)	0.948±0.002	0.950±0.002 (-0.015)	0.955±0.002	0.950±0.001 (-0.006)	0.974±0.001
	Embedding	0.932±0.002 (-0.021)	0.938±0.003	0.944±0.003 (-0.021)	0.948±0.004	0.950±0.002 (-0.006)	0.970±0.003
ACM	Projection	0.867±0.021 (-0.062)	0.868±0.032	0.887±0.016 (-0.048)	0.903±0.009	0.905±0.006 (-0.032)	0.936±0.011
	Prediction	0.886±0.008 (-0.043)	0.889±0.014	0.914±0.009 (-0.021)	0.931±0.008	0.909±0.016 (-0.028)	0.932±0.018
	Embedding	0.884±0.004 (-0.045)	0.874±0.011	0.874±0.009 (-0.061)	0.888±0.011	0.909±0.009 (-0.028)	0.941±0.006
Amazon	Projection	0.679±0.028 (-0.177)	0.676±0.029	0.720±0.049 (-0.233)	0.724±0.050	0.733±0.062 (-0.204)	0.741±0.064
	Prediction	0.811±0.029 (-0.045)	0.778±0.027	0.925±0.008 (-0.028)	0.931±0.006	0.899±0.021 (-0.038)	0.916±0.023
	Embedding	0.890±0.005 (0.034)	0.846±0.018	0.906±0.004 (-0.047)	0.918±0.002	0.925±0.005 (-0.012)	0.941±0.005

(a) t-SNE projection   (b) Prediction   (c) Embedding
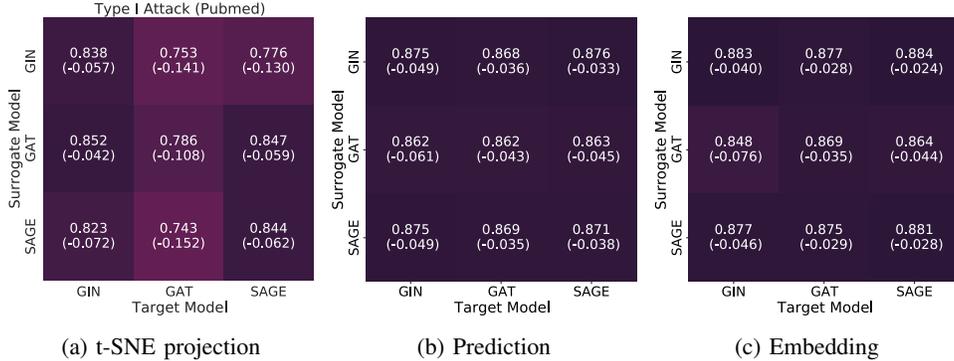
Fig. 2: Heatmap of the accuracy scores of Type I attacks. We show the performance results of 9 combinations of surrogate and target models given different response information. The accuracy differences (in parenthesis) of the surrogate models to the target models are also reported. We fix the dataset to Pubmed.

models given different combinations of GNN architectures for each response. The subtle performance drop only occurs when GAT is the target model and the response is a t-SNE projection, i.e., Type I.3 attack. Even in this case, the adversary can steal usable surrogate models with no more than a 0.162 accuracy score drop, which indicates our attack remains effective. Our results demonstrate that the adversary does not require knowledge about the architecture of target models to conduct the attacks.

To better illustrate it, take the ACM dataset as an example, we extract the embeddings of a given set of nodes from both target and surrogate models and project them into a 2-dimensional space using t-SNE. The result is shown in Figure 3. We use the triangle (cross) to denote the embeddings extracted from the target (surrogate) model and different colors to denote different classes. We find that for both target and surrogate models, the different classes' embeddings can be

separated easily. It means that the surrogate model can also successfully map nodes from different classes into different space, which lead to high accuracy.

Fidelity. The reconnaissance adversary's motivation is faithfully copying the behavior of the target model. We also summarize the fidelity performance of the surrogate models in Table V. It is straightforward to see that the better accuracy performance a surrogate model can reach, the better fidelity it can achieve. For instance, on the Coauthor dataset, the surrogate models reach at least 0.816 accuracy score while these models achieve at least 0.817 fidelity score to the target models in all 9 cases. The surrogate models for other datasets also follow similar patterns. We then calculate the Pearson correlation coefficient between accuracy and fidelity of the surrogate models given three target models. The coefficient scores are 0.957, 0.988, and 0.959 respectively. The results exemplify that the fidelity of the surrogate models to the target
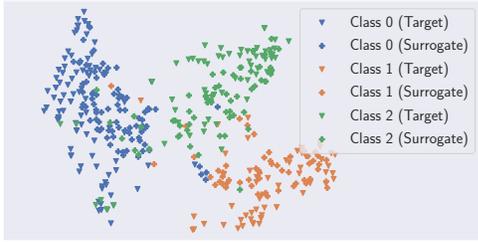
Fig. 3: The embeddings (256-dimension) obtained from the target and surrogate models of 200 randomly selected nodes on the ACM dataset. We project them into a 2-dimensional space using t-SNE. The target model is GAT and the surrogate model is SAGE.

Fig. 4: The accuracy scores of Type II attacks on the Citeseer dataset using different graph reconstruction methods. We use GAT as the target model and GraphSAGE as the surrogate model. The fidelity scores can be found in Appendix Section C.

TABLE VI: The accuracy and fidelity scores of Attack-3 [78] and our Type I attacks on the Pubmed dataset. We use GraphSAGE as the surrogate model.

Method	Accuracy	Fidelity
Attack-3 [78]	0.799	0.818
Our attack (GIN)	0.875	0.903
Our attack (GAT)	0.869	0.898
Our attack (SAGE)	0.871	0.924

models is highly correlated with their accuracy performance. From Figure 3, we can also observe that for each class, the embeddings extracted from target and surrogate models lie in the same region, which implies that the surrogate model has the ability to generate the node embeddings that is close to the one generated by the target model, which leads to high fidelity. Besides, we compare our attacks with Attack-3 proposed by Wu et al. [78] using the overlapping Pubmed dataset and the results are shown in Table VI. Note that we use fewer data to train the surrogate model compared to Wu et al. [78]. Given SAGE as $\mathcal{M}_S$ and GIN, GAT, and SAGE as $\mathcal{M}_T$, our attack achieves 0.903, 0.898, and 0.924 fidelity scores respectively while their attack only achieves 0.818 fidelity score. The results exemplify that interacting with the target model can better facilitate the adversary to attain the reconnaissance goal.

Stability. We run each combination 5 times with different graph partition seeds. It enables us to measure how widely accuracy/fidelity values are dispersed from the average value (i.e., standard deviation). A low standard deviation indicates a low volatility. As we can observe in Table V, the standard deviation values are low in all cases. It shows that the adversary can steal from the target models with statistically stable accuracy and fidelity.

Observation. To achieve high fidelity, the adversary wants to make sure the mistakes and correct labels are the same between the surrogate and target models. When the target model achieves high accuracy, it can make the correct predictions for most of the test data while making few mistakes. If the surrogate model gets close to the accuracy performance of the target model, it would gain high fidelity to the target model due to the fact the number of correctly predicted instances by
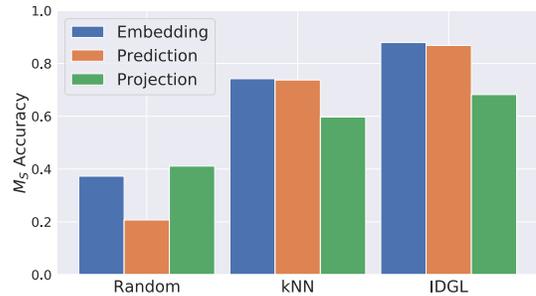
the surrogate model would considerably overlap those made by the target model. The above Pearson correlation results verify this intuitive explanation.

Takeaways. The theft adversary can reliably build accurate surrogate models close to the target models via Type I attacks. It is worth noting that, in the real world, it is hard for the reconnaissance adversary to comprehensively verify the fidelity of the surrogate models without risking a number of queries. Our results imply that the reconnaissance adversary may focus on building surrogate models that preserve high accuracy similar to the target models. In turn, these surrogate models would likely be faithful to the remote targets. Besides, our results demonstrate that the adversary does not require knowledge of the target models' architectures.

C. Performance Evaluation: Type II Attacks

To launch Type II attacks, the adversary first builds an adjacency matrix $\mathbf{A}_Q$ for query graph $\mathbf{G}_Q$ before querying the target model $\mathcal{M}_T$. We follow the aforementioned graph reconstruction configuration to restore $\mathbf{A}_Q$ for query graph $\mathbf{G}_Q$ and conduct the model stealing attacks. The performance of Type II attacks is summarized in Table VII. Due to space limitations, we only show the attack results when the adversary uses GraphSAGE as the surrogate model. The performance results using GIN and GAT as the surrogate models follow similar patterns and can be found in Appendix Section B.

Accuracy. As we can see in Table VII, given all the datasets, the adversary can launch Type II.1/2 attacks to build surrogate models that offer accuracy on par with the target models. At the same time, we observe that the adversary can launch Type II.3 attacks and steal usable surrogate models. Take the Pubmed dataset as an example, when the response is the t-SNE projection of query nodes, the surrogate models achieve the average accuracy score of 0.836, 0.739, and 0.850 concerning different target models. This represents 0.166 accuracy drop in the worst case (compared to the target model performance). In general, Type II.3 attacks can achieve comparable accuracy performance in all 6 datasets. Also, we investigate if the adversary can build accurate surrogate models given different

TABLE VII: The accuracy and fidelity scores of Type II attacks using different response information on all the 6 datasets. Both average values and standard deviations are reported. The accuracy differences (in parenthesis) of the surrogate models to the target models are also reported. We use GraphSAGE as the surrogate model.

Dataset	$\mathcal{M}_S$ (SAGE)	$\mathcal{M}_T$					
		GIN		GAT		SAGE	
		Accuracy	Fidelity	Accuracy	Fidelity	Accuracy	Fidelity
DBLP	Projection	0.703±0.018 (-0.169)	0.732±0.018	0.675±0.007 (-0.163)	0.693±0.009	0.713±0.024 (-0.145)	0.761±0.028
	Prediction	0.779±0.008 (-0.093)	0.824±0.009	0.782±0.005 (-0.056)	0.832±0.004	0.809±0.004 (-0.049)	0.882±0.004
	Embedding	0.783±0.009 (-0.089)	0.825±0.010	0.787±0.004 (-0.051)	0.834±0.009	0.812±0.004 (-0.046)	0.886±0.003
Pubmed	Projection	0.836±0.015 (-0.088)	0.862±0.015	0.739±0.018 (-0.166)	0.738±0.022	0.850±0.017 (-0.059)	0.893±0.020
	Prediction	0.878±0.006 (-0.046)	0.908±0.003	0.867±0.003 (-0.038)	0.897±0.003	0.872±0.004 (-0.037)	0.928±0.004
	Embedding	0.879±0.004 (-0.045)	0.910±0.005	0.875±0.003 (-0.030)	0.905±0.002	0.881±0.002 (-0.028)	0.941±0.002
Citeseer	Projection	0.647±0.028 (-0.263)	0.658±0.024	0.682±0.030 (-0.229)	0.679±0.033	0.695±0.021 (-0.224)	0.712±0.022
	Prediction	0.834±0.012 (-0.076)	0.862±0.011	0.868±0.005 (-0.043)	0.896±0.004	0.880±0.007 (-0.039)	0.928±0.007
	Embedding	0.827±0.009 (-0.083)	0.850±0.014	0.879±0.006 (-0.032)	0.904±0.008	0.880±0.006 (-0.039)	0.923±0.010
Coauthor	Projection	0.862±0.052 (-0.091)	0.866±0.054	0.864±0.044 (-0.101)	0.867±0.043	0.793±0.044 (-0.163)	0.801±0.046
	Prediction	0.955±0.001 (0.002)	0.960±0.001	0.954±0.001 (-0.011)	0.958±0.001	0.954±0.002 (-0.002)	0.976±0.002
	Embedding	0.950±0.002 (-0.003)	0.955±0.001	0.947±0.003 (-0.018)	0.950±0.002	0.951±0.002 (-0.005)	0.965±0.006
ACM	Projection	0.854±0.036 (-0.075)	0.842±0.035	0.888±0.012 (-0.047)	0.904±0.012	0.918±0.007 (-0.019)	0.947±0.008
	Prediction	0.911±0.007 (-0.018)	0.899±0.018	0.927±0.004 (-0.008)	0.934±0.006	0.923±0.005 (-0.014)	0.944±0.007
	Embedding	0.887±0.012 (-0.042)	0.877±0.010	0.876±0.005 (-0.059)	0.889±0.012	0.921±0.009 (-0.016)	0.944±0.010
Amazon	Projection	0.640±0.020 (-0.216)	0.640±0.021	0.691±0.015 (-0.262)	0.662±0.019	0.739±0.034 (-0.198)	0.743±0.033
	Prediction	0.802±0.031 (-0.054)	0.767±0.026	0.884±0.005 (-0.069)	0.890±0.008	0.922±0.009 (-0.015)	0.931±0.007
	Embedding	0.741±0.109 (-0.115)	0.734±0.117	0.892±0.018 (-0.061)	0.900±0.019	0.938±0.005 (0.001)	0.954±0.005

(a) t-SNE Projection (b) Prediction (c) Embedding
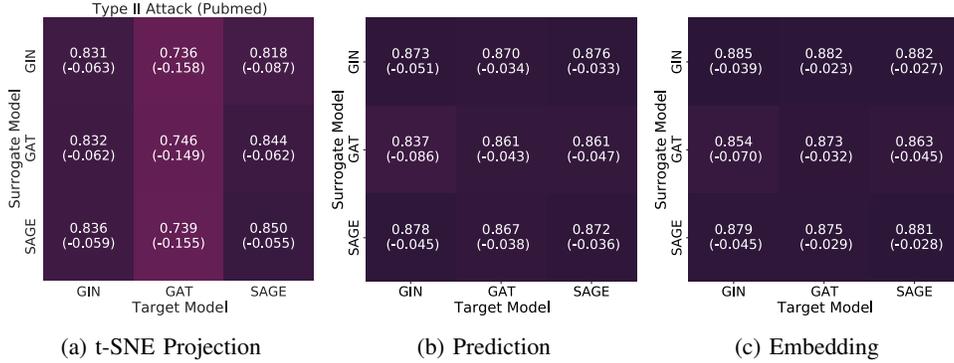
Fig. 5: Heatmap of the accuracy scores of Type II attacks. We show the performance results of 9 combinations of surrogate and target models given different response information. The accuracy differences (in parenthesis) of the surrogate models to the target models are also reported. We fix the dataset to Pubmed.

combinations of GNN architectures for each response in Type II attacks. As we can observe from Figure 5, when the response is predicted posterior probability or embedding, the accuracy of the surrogate model is close to the target model. Regarding the case when the response is t-SNE projection, our attack still works well. We only witness a slight performance drop when the target model is GAT. The results exemplify that, in general, Type II attacks remain effective in stealing target models with different architectures.

Fidelity. We also observe the same correlation between accuracy and fidelity in Type II attacks. That is, the better accuracy performance a surrogate model can reach, the better fidelity it can achieve. Take the Coauthor dataset as an example, the surrogate models reach at least 0.793 accuracy score while these models achieve at least 0.801 fidelity score to the target models in all cases. We also calculate the Pearson correlation coefficient between accuracy and fidelity of the surrogate

models given three target models. The coefficient scores are 0.970, 0.991, and 0.967 respectively. These correlation scores are similar to what we observe from Type I attack results.

Efficacy of Learned Query Graph Structure. To investigate the effect of graph structure, we compare IDGL with two additional methods, i.e., random graph construction and kNN. For the two additional methods, we set the average node degree to 24, which is the same value used to initialize IDGL. Due to space limitations, we only show the accuracy of Type II attacks on the Citeseer dataset using GAT as the target model and GraphSAGE as the surrogate model. The accuracy and fidelity of other datasets follow similar patterns. As we can see from Figure 4, using IDGL to reconstruct the graph structure reaches the highest attack accuracy for all responses. For instance, the accuracy score is 0.879 when using IDGL as the reconstruction method and taking embedding as the response, while the corresponding accuracy score is only 0.411
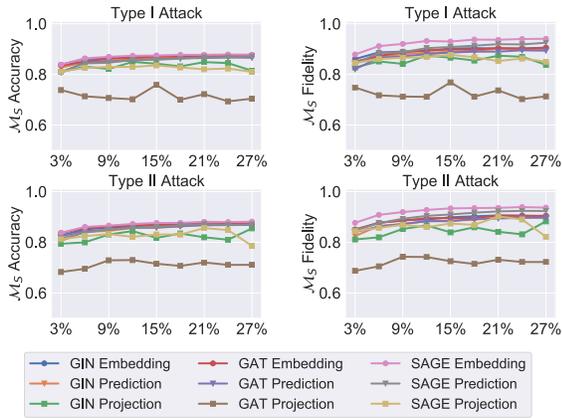
Fig. 6: The average accuracy (fidelity) scores of Type I (Type II) attacks using different response information on the Pubmed datasets. We use GraphSAGE as the surrogate model. The x-axis represents the percentage of randomly selected nodes in the dataset and the y-axis represents the accuracy (fidelity) of surrogate models.

(a) t-SNE Projection   (b) Embedding

Fig. 7: Defense: adding Gaussian noise to t-SNE projection and node embedding. We report the accuracy scores and standard deviation of the surrogate model (GraphSAGE) given all three target models under this defense mechanism (Type I.1 and I.3 attacks, respectively). σ is the standard deviation of the Gaussian noise. We fix the dataset to ACM.

and 0.737 respectively when using random graph construction or kNN as the reconstruction method. It demonstrates that an effective graph reconstruction method does benefit the final attack performance.

Stability. As we can observe in Table VII, the standard deviation values remain low in all cases. This shows that the adversary can steal the target models with statistically stable accuracy and fidelity in Type II attacks.

Observation. When comparing Table VII to Table V, we observe that Type II attack achieves better performance than Type I attack in certain cases. Chen et al. [11] observed a similar phenomenon and concluded that the raw graphs are not always optimal for the downstream tasks for different reasons. For example, raw graphs may contain noisy/incomplete information due to the error-prone data collection or their structures do not reflect the ideal graph topology after feature extraction and transformation. The query graphs learned by the IDGL framework in our Type II attacks are optimized toward the downstream tasks (e.g., node classification) and may achieve better performance in some cases. We refer the audience to Chen et al. [11] for additional details.

Takeaways. Our results show that the attack framework enables the adversary to learn a discrete graph structure and steal usable surrogate models. Coupling with the results shown in Section V-B, we demonstrate that our model stealing attacks achieve strong performance concerning different responses.

D. Performance Evaluation: Query Budget

We then investigate the attack performance with respect to different query budgets, i.e., different sizes of query graph $\mathbf{G}_Q$. Due to space limitations, we only show the results for the Pubmed dataset, other datasets follow similar trends. The corresponding accuracy and fidelity of Type I and II attacks are summarized in Figure 6. We observe that in general, larger
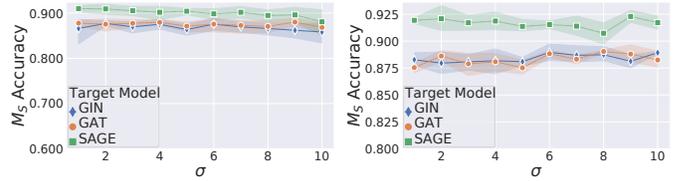
query budgets lead to better accuracy and fidelity. For instance, in Type I attack, when the response is GraphSAGE's embedding, the accuracy score increases from 0.839 to 0.870 when the query budget increases from 3% to 27%. However, in most of the cases, we can achieve similar performance even using only 3% of randomly sampled nodes of the original dataset (10% of the $\mathbf{G}_Q$ used in previous experiments). The results demonstrate that the adversary can still launch effective attacks even with a low-quality query graph (less query budget and no graph structure information). Besides, when the query budget is small, the adversary can investigate the characteristics of query data (e.g., sparsity of the learned graph) and decide if edge reconstruction is necessary to launch attacks.

VI. DISCUSSIONS

Limitation. Our model stealing attack is limited to the scope that the target model returns node-level results. We did not explore the scenario when the target model accepts an arbitrary graph as input and returns graph-level results. That is, the target models represent the whole structure of graphs using various pooling methods and return a single embedding vector for downstream graph-level tasks such as graph classification [82], [85]. Model stealing attack under this setting would require a different approach. Besides, we do not jointly optimize both graph structure and surrogate model. Such training paradigm may cost more training epochs to converge, and the query budget would also increase, which inevitably increases the risk of being detected. We put them into our future work.

Defense. Several countermeasures to model stealing attacks have been discussed in previous literature [52], [70]. One straightforward countermeasure is injecting perturbations to the predicted posterior probability reported by the classifier, i.e., perturb the probability while retaining the top-1 label [44], [70]. To cope with different types of query responses, we consider adding random noise to the response regardless of the corresponding label, i.e. the distribution of the random noise is independent of the node class label. Concretely, we add random Gaussian noise into node embedding and t-SNE projection returned by all three target models and use GraphSAGE as the surrogate model to understand the

effectiveness of such countermeasure. According to the defined threat model, the accuracy of the surrogate model measures the attack performances. A higher accuracy of the surrogate model indicates a more successful attack. We use the ACM dataset as an example and the results are shown in Figure 7. We observe that the random Gaussian noise slightly affects the accuracy performance of the surrogate model. For instance, in Figure 7a, when σ (i.e., the standard deviation of the added noise) is greater than 7, we can observe the accuracy performance of the surrogate model starts to decrease for different target models. In contrast, if the query response from the target model is the node embedding vector, we can only observe much fewer fluctuations of the surrogate model's accuracy with increasingly stronger random Gaussian noise injected to the embedding (see Figure 7b). To summarize, our preliminary experiment does not establish concrete evidence that adding random noise would counter our attacks. We leave designing effective defense mechanisms for model stealing attacks against GNNs as our future work.

VII. RELATED WORK

In this section, we review the research work close to our proposed attacks. We refer the readers to [23], [42], [81], [91], [92] for an in-depth overview of different GNN models, and [10], [14], [37], [66], [83] for comprehensive surveys of existing adversarial attacks and defense strategies on GNNs.

Model Stealing Attack Against ML Models. Model extraction is in many ways similar to model distillation, but it differs in that the victim's proprietary training set is not accessible to the adversary. In this regard, previous literature already investigated stealing various aspects of a black-box ML model such as hyperparameters [74], architecture [51], information on training data [27], [28], parameters [4], [70], decision boundaries [54], and functionality [34], [52]. However, most of these efforts focused on images. There exist some preliminary work on model stealing attacks against GNNs [16], [78]. However, they are only focusing on the transductive setting of GNNs, which cannot generalize to unseen data. Our attacks instead focus on a more popular and general setting of GNNs, i.e., inductive setting. We fill the gap and understand if both theft and reconnaissance adversaries can steal inductive GNNs with high accuracy and high fidelity.

Causative Attacks on GNNs. Many adversarial attacks to GNNs [2], [14], [19], [49], [67], [68], [84], [89], [90], [97] are causative attacks [31]. These attacks assume that an adversary can manipulate the training dataset in order to change the parameters of the target model and influence their behavior. In this context, Zügner et al. [96] was the first to introduce unnoticeable adversarial perturbations of the node's features and the graph structure. Their goal was to reduce the accuracy of node classification via GCN. After this work, different adversarial attack strategies have been proposed. Depending on the attack objectives, they aim at reducing the accuracy of node classification [2], [7], [49], [68], [84], [97] (node level), link prediction [2], [9], [46], [67] (edge level), graph classification [14], [89] (graph level), etc. Our attack does not

tamper with the training graph data and does not change the behavior of the target model or its parameters.

Exploratory Attacks on GNNs. In reality, however, it is more practical for the attacker to query the target model and leverage the model's responses on these carefully crafted input data. Consequently, we witness the emerging of exploratory attacks on ML models. However, adversarial exploratory attacks on GNNs remain understudied. In particular, only a few studies [18], [27], [78] focused on exploratory attacks on GNNs. For instance, He et al. [27] proposed the first link stealing attack to infer if there exists an edge between a given pair of nodes in the training graph. Duddu et al. [18] and He et al. [28] discussed the membership inference attack that infers whether a given node in the graph was used to train the target model by leveraging different background knowledge. Note that, in membership inference attacks, the attack model in its training phase does not interact with the target model. However, in model stealing attacks, the surrogate model does have interaction with the target model since it needs the guidance of the target model to optimize its parameters.

Defense of Attacks on GNNs. To mitigate those attacks, several defense strategies have been proposed. The core idea of the existing defense strategies is reducing the sensitivity of GNNs using adversarial training [15], [17], [20], [36], [65], [77], perturbation detection [33], graph sanitization [79], [88], etc. In turn, the trained GNNs are robust to perturbation (e.g., structure perturbation [15], [36], [77], attribution perturbation [17], [20], [36], [65], [77]). Also, robustness certification [3], [98] becomes an emerging research direction. They aim at reasoning the safety posture of GNNs under adversarial perturbations. However, these defense techniques only protect GNNs from causative attacks instead of exploratory attacks.

VIII. CONCLUSION

In this paper, we perform the first security risk assessment against inductive GNNs through the lens of model stealing attacks. We propose a threat model to systematically categorize an adversary's background knowledge into two dimensions, i.e., query graph and model responses. By jointly considering the two dimensions, we summarize six attack scenarios. We then propose a general attack framework that can be applied in different scenarios. Extensive experiments on three popular inductive GNN architectures and six benchmark datasets show that our model stealing attacks can handle different types of responses and achieve strong performance. Moreover, the attacks are still effective even the adversary has no knowledge about the graph structural information.

REFERENCES

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," in *International Conference on Learning Representations (ICLR)*, 2015.

[2] 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.

[3] A. Bojchevski and S. Günnemann, "Certifiable Robustness to Graph Perturbations," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2019, pp. 8317–8328.

[4] N. Carlini, M. Jagielski, and I. Mironov, "Cryptanalytic Extraction of Neural Network Models," in *Annual International Cryptology Conference (CRYPTO)*. Springer, 2020, pp. 189–218.

[5] N. Carlini, F. Tramèr, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. B. Brown, D. Song, Ú. Erlingsson, A. Oprea, and C. Raffel, "Extracting Training Data from Large Language Models," *CoRR abs/2012.07805*, 2020.

[6] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2017, pp. 39–57.

[7] J. Caverlee, X. B. Hu, M. Lalmas, and W. Wang, "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.

[8] D. Chen, N. Yu, Y. Zhang, and M. Fritz, "GAN-Leaks: A Taxonomy of Membership Inference Attacks against Generative Models," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2020, pp. 343–362.

[9] J. Chen, X. Lin, Z. Shi, and Y. Liu, "Link Prediction Adversarial Attack Via Iterative Gradient Attack," *IEEE Transactions on Computational Social Systems*, 2020.

[10] L. Chen, J. Li, J. Peng, T. Xie, Z. Cao, K. Xu, X. He, and Z. Zheng, "A Survey of Adversarial Learning on Graphs," *CoRR abs/2003.05730*, 2020.

[11] Y. Chen, L. Wu, and M. J. Zaki, "Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[12] Z. Chen, F. Chen, L. Zhang, T. Ji, K. Fu, L. Zhao, F. Chen, and C. Lu, "Bridging the Gap between Spatial and Spectral Domains: A Survey on Graph Neural Networks," *CoRR abs/2002.11867*, 2020.

[13] C. A. C. Choo, F. Tramèr, N. Carlini, and N. Papernot, "Label-Only Membership Inference Attacks," in *International Conference on Machine Learning (ICML)*. PMLR, 2021, pp. 1964–1974.

[14] 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.

[15] Q. Dai, X. Shen, L. Zhang, Q. Li, and D. Wang, "Adversarial Training Methods for Network Embedding," in *The Web Conference (WWW)*. ACM, 2019, pp. 329–339.

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

[17] Z. Deng, Y. Dong, and J. Zhu, "Batch Virtual Adversarial Training for Graph Convolutional Networks," *CoRR abs/1902.09192*, 2019.

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

[19] 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.

[20] F. Feng, X. He, J. Tang, and T. Chua, "Graph Adversarial Training: Dynamically Regularizing Based on Graph Structure," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[21] C. L. Giles, K. D. Bollacker, and S. Lawrence, "CiteSeer: An Automatic Citation Indexing System," in *International Conference on Digital Libraries (ICDL)*. ACM, 1998, pp. 89–98.

[22] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," in *International Conference on Learning Representations (ICLR)*, 2015.

[23] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge Based Systems*, 2018.

[24] 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.

[25] 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.

[26] C. He, K. Balasubramanian, E. Ceyani, Y. Rong, P. Zhao, J. Huang, M. Annavaram, and S. Avestimehr, "FedGraphNN: A Federated Learning System and Benchmark for Graph Neural Networks," *CoRR abs/2104.07145*, 2021.

[27] 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.

[28] 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.

[29] X. He and Y. Zhang, "Quantifying and Mitigating Privacy Risks of Contrastive Learning," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021, pp. 845–863.

[30] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, and J. Leskovec, "Strategies for Pre-training Graph Neural Networks," in *International Conference on Learning Representations (ICLR)*, 2020.

[31] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, "Adversarial Machine Learning," in *Workshop on Security and Artificial Intelligence (AISec)*. ACM, 2011, pp. 43–58.

[32] Y. Huang, J. Zhang, Y. Yang, Z. Gong, and Z. Hao, "GNNVis: Visualize Large-Scale Data by Learning a Graph Neural Network Representation," in *ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 2020, pp. 545–554.

[33] V. N. Ioannidis, D. Berberidis, and G. B. Giannakis, "GraphSAC: Detecting Anomalies in Large-scale Graphs," *CoRR abs/1910.09589*, 2019.

[34] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, "High Accuracy and High Fidelity Extraction of Neural Networks," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2020, pp. 1345–1362.

[35] J. Jia, A. Salem, M. Backes, Y. Zhang, and N. Z. Gong, "MemGuard: Defending against Black-Box Membership Inference Attacks via Adversarial Examples," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2019, pp. 259–274.

[36] H. Jin and X. Zhang, "Latent Adversarial Training of Graph Convolution Networks," in *ICML Workshop on Learning and Reasoning with Graph-Structured Representations (LRG)*. ICML, 2020.

[37] W. Jin, Y. Li, H. Xu, Y. Wang, and J. Tang, "Adversarial Attacks and Defenses on Graphs: A Review and Empirical Study," *CoRR abs/2003.00653*, 2020.

[38] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "PRADA: Protecting Against DNN Model Stealing Attacks," in *IEEE European Symposium on Security and Privacy (Euro S&P)*. IEEE, 2019, pp. 512–527.

[39] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular Graph Convolutions: Moving Beyond Fingerprints," *Journal of Computer-Aided Molecular Design*, 2016.

[40] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[41] K. Krishna, G. S. Tomar, A. P. Parikh, N. Papernot, and M. Iyyer, "Thieves on Sesame Street! Model Extraction of BERT-based APIs," in *International Conference on Learning Representations (ICLR)*, 2020.

[42] J. B. Lee, R. A. Rossi, S. Kim, N. K. Ahmed, and E. Koh, "Attention Models in Graphs: A Survey," *ACM Transactions on Knowledge Discovery from Data*, 2019.

[43] J. Lee, I. Lee, and J. Kang, "Self-Attention Graph Pooling," in *International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 3734–3743.

[44] T. Lee, B. Edwards, I. Molloy, and D. Su, "Defending Against Neural Network Model Stealing Attacks Using Deceptive Perturbations," in *IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 43–49.

[45] Z. Li and Y. Zhang, "Membership Leakage in Label-Only Exposures," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2021, pp. 880–895.

[46] W. Lin, S. Ji, and B. Li, "Adversarial Attacks on Link Prediction Algorithms Based on Graph Neural Networks," in *ACM Asia Conference on Computer and Communications Security (ASIACCS)*. ACM, 2020, pp. 370–380.

[47] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and Q. He, "Pick and Choose: A GNN-based Imbalanced Learning Approach for Fraud Detection," in *The Web Conference (WWW)*. ACM, 2021, pp. 3168–3177.

[48] Y. Liu, R. Wen, X. He, A. Salem, Z. Zhang, M. Backes, E. D. Cristofaro, M. Fritz, and Y. Zhang, "ML-Doctor: Holistic Risk Assessment of Inference Attacks Against Machine Learning Models," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2022.

[49] 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.

[50] 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.

[51] S. J. Oh, M. Augustin, B. Schiele, and M. Fritz, "Towards Reverse-Engineering Black-Box Neural Networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[52] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff Nets: Stealing Functionality of Black-Box Models," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 4954–4963.

[53] 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.

[54] N. Papernot, P. D. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical Black-Box Attacks Against Machine Learning," in *ACM Asia Conference on Computer and Communications Security (ASIACCS)*. ACM, 2017, pp. 506–519.

[55] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2016, pp. 582–597.

[56] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online Learning of Social Representations," in *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2014, pp. 701–710.

[57] Y. Rong, Y. Bian, T. Xu, W. Xie, Y. Wei, W. Huang, and J. Huang, "Self-Supervised Graph Transformer on Large-Scale Molecular Data," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2020.

[58] A. Sablayrolles, M. Douze, C. Schmid, Y. Ollivier, and H. Jégou, "White-box vs Black-box: Bayes Optimal Strategies for Membership Inference," in *International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 5558–5567.

[59] 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.

[60] J. Scott, *Social Network Analysis*. SAGE Publications Inc, 2017.

[61] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective Classification in Network Data," *AI Magazine*, 2008.

[62] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of Graph Neural Network Evaluation," *CoRR abs/1811.05868*, 2018.

[63] 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.

[64] C. Song and V. Shmatikov, "Auditing Data Provenance in Text-Generation Models," in *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2019, pp. 196–206.

[65] K. Sun, Z. Lin, H. Guo, and Z. Zhu, "Virtual Adversarial Training on Graph Convolutional Networks in Node Classification," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*. Springer, 2019, pp. 431–443.

[66] L. Sun, Y. Dou, C. Yang, J. Wang, P. S. Yu, L. He, and B. Li, "Adversarial Attack and Defense on Graph Data: A Survey," *CoRR abs/1812.10528*, 2018.

[67] M. Sun, J. Tang, H. Li, B. Li, C. Xiao, Y. Chen, and D. Song, "Data Poisoning Attack against Unsupervised Node Embedding Methods," *CoRR abs/1810.12881*, 2018.

[68] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, "Non-target-specific Node Injection Attacks on Graph Neural Networks: A Hierarchical Reinforcement Learning Approach," in *The Web Conference (WWW)*. ACM, 2020, pp. 673–683.

[69] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble Adversarial Training: Attacks and Defenses," in *International Conference on Learning Representations (ICLR)*, 2017.

[70] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing Machine Learning Models via Prediction APIs," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2016, pp. 601–618.

[71] R. van den Berg, T. N. Kipf, and M. Welling, "Graph Convolutional Matrix Completion," *CoRR abs/1706.02263*, 2017.

[72] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, 2008.

[73] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[74] B. Wang and N. Z. Gong, "Stealing Hyperparameters in Machine Learning," in *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2018, pp. 36–52.

[75] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang, "Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks," *CoRR abs/1909.01315*, 2019.

[76] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous Graph Attention Network," in *The Web Conference (WWW)*. ACM, 2019, pp. 2022–2032.

[77] X. Wang, X. Liu, and C. Hsieh, "GraphDefense: Towards Robust Graph Convolutional Networks," *CoRR abs/1911.04429*, 2019.

[78] B. Wu, X. Yang, S. Pan, and X. Yuan, "Model Extraction Attacks on Graph Neural Networks: Taxonomy and Realization," *CoRR abs/2010.12751*, 2020.

[79] 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.

[80] L. Wu, Y. Yang, L. Chen, D. Lian, R. Hong, and M. Wang, "Learning to Transfer Graph Embeddings for Inductive Graph based Recommendation," in *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 2020, pp. 1211–1220.

[81] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[82] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph Backdoor," in *USENIX Security Symposium (USENIX Security)*. USENIX, 2021.

[83] H. Xu, Y. Ma, H. Liu, D. Deb, H. Liu, J. Tang, and A. K. Jain, "Adversarial Attacks and Defenses in Images, Graphs and Text: A Review," *International Journal of Automation and Computing*, 2020.

[84] 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.

[85] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?" in *International Conference on Learning Representations (ICLR)*, 2019.

[86] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," *ACM Transactions on Intelligent Systems and Technology*, 2019.

[87] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical Graph Representation Learning with Differentiable Pooling," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*. NeurIPS, 2018, pp. 4805–4815.

[88] Y. Zhang, S. H. Khan, and M. Coates, "Comparing And Detecting Adversarial Attacks for Graph Deep Learning," in *ICLR Representation Learning on Graphs and Manifolds Workshop (RLGM)*, 2019.

[89] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong, "Backdoor Attacks to Graph Neural Networks," *CoRR abs/2006.11165*, 2020.

[90] 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.

[91] Z. Zhang, P. Cui, and W. Zhu, "Deep Learning on Graphs: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[92] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, "Graph Neural Networks: A Review of Methods and Applications," *CoRR abs/1812.08434*, 2018.
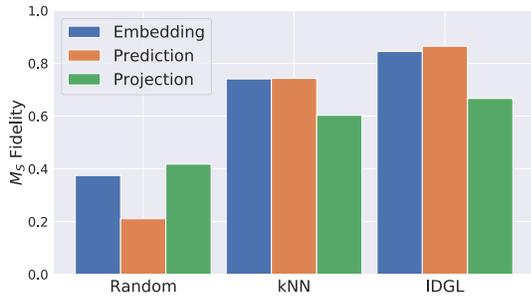
Fig. 8: The fidelity score of Type II attacks on the Citeseer dataset using different graph reconstruction methods. We use GAT as the target model and GraphSAGE as the surrogate model.

[93] M. Zhu, X. Wang, C. Shi, H. Ji, and P. Cui, "Interpreting and Unifying Graph Neural Networks with An Optimization Framework," in *The Web Conference (WWW)*. ACM, 2021, pp. 1215–1226.

[94] Q. Zhu, Y. Xu, H. Wang, C. Zhang, J. Han, and C. Yang, "Transfer Learning of Graph Neural Networks with Ego-graph Information Maximization," *CoRR abs/2009.05204*, 2020.

[95] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, 2018.

[96] 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.

[97] D. Zügner and S. Günnemann, "Adversarial Attacks on Graph Neural Networks via Meta Learning," in *International Conference on Learning Representations (ICLR)*, 2019.

[98] D. Zügner and S. Günnemann, "Certifiable Robustness and Robust Training for Graph Convolutional Networks," in *ACM Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2019, pp. 246–256.

APPENDIX

A. Performance Evaluation: Type I attacks (GIN/GAT)

The performance of Type I attacks using GIN and GAT as surrogate models are summarized in Table VIII and Table IX.

B. Performance Evaluation: Type II attacks (GIN/GAT)

The performance of Type II attacks using GIN and GAT as surrogate models are summarized in Table X and Table XI.

C. Fidelity

The fidelity score of Type II attacks on the Citeseer dataset using different graph reconstruction methods is shown in Figure 8.

D. Hyperparameter Study

GNNs are complex and their performance may be affected by the hyperparameter settings. This is especially important for our model stealing attack since we use inductive GNNs as the part of our surrogate models. At the same time, our attack is in a full black-box setting. As such, we evaluate the impact of three hyperparameters on the performance of surrogate models, i.e., hidden unit size, number of epochs, and batch size. For each hyperparameter, we only show the results on one dataset under Type I attacks given the space limitation. Other datasets follow a similar trend.

Hidden Unit Size. In general, the larger the hidden units, the greater space of representation functions a graph convolutional
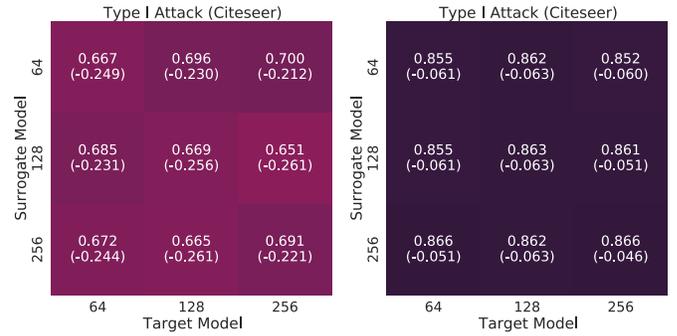
(a) t-SNE Projection   (b) Prediction

Fig. 9: Hyperparameter study: hidden unit size. We show the influence of different combinations of the hidden unit size on the surrogate model's performance.
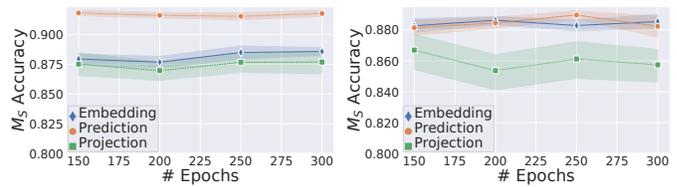
(a) Target model (GAT)   (b) Target model (GIN)

Fig. 10: Hyperparameter study: number of epochs. We report the accuracy and standard deviation of the surrogate model (GraphSAGE) under Type I attacks given different number of epochs. We fix the dataset to ACM.
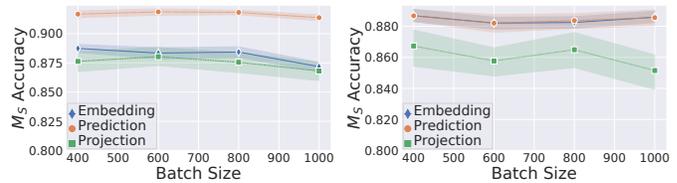
(a) Target model (GAT)   (b) Target model (GIN)

Fig. 11: Hyperparameter study: batch size. We report the accuracy and standard deviation of the surrogate model (Graph-SAGE) under Type I attacks given different batch sizes. We fix the dataset to ACM.

layer can offer. As such, we investigate how the hidden unit size may affect the attack performance. Recall that the adversary does not know the target model's hidden unit size. They can only blindly guess the hidden unit size used by the target model. To this end, we first build a target model (i.e., GAT) with 64, 128, and 256 hidden units using Citeseer dataset. We then launch Type I attacks using a fixed surrogate model (i.e., GraphSAGE) with 64, 128, and 256 hidden units. In this way, we can observe the potential performance changes in different circumstances. The results are shown in Figure 9a and Figure 9b, respectively. We can see that different hidden unit sizes adopted by the surrogate models have a limited impact on the accuracy performance. Our hypothesis is that

TABLE VIII: The accuracy and fidelity score of Type I attacks using different response information on all the 6 datasets. Both average values and standard deviations are reported. The accuracy differences (in parenthesis) of the surrogate models to the target models are also reported. We use GIN as the surrogate model.

| Dataset | $\mathcal{M}_S$ (GIN) | $\mathcal{M}_T$ | | | | | |
| | | GIN | | GAT | | SAGE | |
		Accuracy	Fidelity	Accuracy	Fidelity	Accuracy	Fidelity
DBLP	Projection	0.710±0.028 (-0.162)	0.737±0.031	0.677±0.005 (-0.161)	0.696±0.008	0.719±0.033 (-0.139)	0.761±0.037
	Prediction	0.751±0.003 (-0.121)	0.788±0.008	0.775±0.004 (-0.063)	0.826±0.004	0.819±0.002 (-0.039)	0.897±0.001
	Embedding	0.743±0.005 (-0.129)	0.769±0.003	0.792±0.004 (-0.046)	0.836±0.004	0.805±0.003 (-0.053)	0.874±0.007
Pubmed	Projection	0.838±0.027 (-0.086)	0.861±0.030	0.753±0.015 (-0.152)	0.767±0.015	0.776±0.058 (-0.133)	0.813±0.066
	Prediction	0.875±0.004 (-0.049)	0.905±0.005	0.868±0.004 (-0.037)	0.896±0.005	0.876±0.003 (-0.033)	0.932±0.002
	Embedding	0.883±0.004 (-0.041)	0.912±0.003	0.877±0.004 (-0.028)	0.903±0.003	0.884±0.001 (-0.025)	0.937±0.004
Citeseer	Projection	0.675±0.018 (-0.236)	0.658±0.016	0.715±0.012 (-0.196)	0.701±0.014	0.733±0.011 (-0.186)	0.727±0.009
	Prediction	0.808±0.005 (-0.103)	0.816±0.006	0.878±0.004 (-0.033)	0.909±0.002	0.895±0.001 (-0.024)	0.942±0.002
	embedding	0.791±0.005 (-0.120)	0.806±0.007	0.889±0.005 (-0.022)	0.913±0.005	0.881±0.006 (-0.038)	0.917±0.006
Coauthor	Projection	0.824±0.038 (-0.129)	0.826±0.039	0.813±0.017 (-0.152)	0.816±0.018	0.796±0.049 (-0.160)	0.806±0.051
	Prediction	0.920±0.002 (-0.033)	0.925±0.003	0.945±0.001 (-0.020)	0.948±0.002	0.956±0.002 (-0.000)	0.976±0.001
	Embedding	0.900±0.005 (-0.053)	0.907±0.007	0.944±0.001 (-0.021)	0.945±0.001	0.947±0.007 (-0.009)	0.962±0.007
ACM	Projection	0.824±0.020 (-0.105)	0.819±0.023	0.835±0.044 (-0.100)	0.849±0.048	0.892±0.018 (-0.045)	0.913±0.022
	Prediction	0.829±0.014 (-0.100)	0.826±0.015	0.916±0.008 (-0.019)	0.925±0.009	0.918±0.007 (-0.019)	0.946±0.004
	Embedding	0.860±0.010 (-0.069)	0.859±0.019	0.920±0.006 (-0.015)	0.930±0.004	0.917±0.014 (-0.020)	0.944±0.014
Amazon	Projection	0.717±0.037 (-0.139)	0.726±0.024	0.719±0.021 (-0.234)	0.725±0.020	0.764±0.028 (-0.173)	0.779±0.030
	Prediction	0.893±0.006 (0.037)	0.848±0.016	0.930±0.008 (-0.023)	0.935±0.007	0.933±0.004 (-0.004)	0.943±0.005
	Embedding	0.897±0.005 (0.041)	0.830±0.021	0.929±0.006 (-0.024)	0.935±0.006	0.931±0.005 (-0.006)	0.934±0.006

TABLE IX: The accuracy and fidelity score of Type I attacks using different response information on all the 6 datasets. Both average values and standard deviations are reported. The accuracy differences (in parenthesis) of the surrogate models to the target models are also reported. We use GAT as the surrogate model.

| Dataset | $\mathcal{M}_S$ (GAT) | $\mathcal{M}_T$ | | | | | |
| | | GIN | | GAT | | SAGE | |
		Accuracy	Fidelity	Accuracy	Fidelity	Accuracy	Fidelity
DBLP	Projection	0.723±0.007 (-0.149)	0.754±0.008	0.671±0.010 (-0.167)	0.689±0.013	0.746±0.007 (-0.112)	0.793±0.005
	Prediction	0.745±0.006 (-0.127)	0.784±0.008	0.782±0.004 (-0.056)	0.834±0.005	0.809±0.005 (-0.049)	0.880±0.005
	Embedding	0.696±0.010 (-0.176)	0.722±0.007	0.796±0.006 (-0.042)	0.844±0.006	0.762±0.039 (-0.096)	0.819±0.048
Pubmed	Projection	0.852±0.003 (-0.072)	0.887±0.003	0.786±0.026 (-0.119)	0.805±0.025	0.847±0.004 (-0.062)	0.893±0.005
	Prediction	0.862±0.000 (-0.062)	0.899±0.002	0.862±0.002 (-0.043)	0.899±0.003	0.863±0.004 (-0.046)	0.921±0.004
	Embedding	0.848±0.006 (-0.076)	0.873±0.006	0.869±0.002 (-0.036)	0.906±0.003	0.864±0.012 (-0.045)	0.911±0.018
Citeseer	Projection	0.743±0.008 (-0.168)	0.746±0.007	0.755±0.003 (-0.156)	0.763±0.009	0.786±0.009 (-0.133)	0.798±0.011
	Prediction	0.798±0.008 (-0.113)	0.808±0.013	0.880±0.004 (-0.031)	0.908±0.007	0.894±0.004 (-0.025)	0.935±0.005
	Embedding	0.776±0.013 (-0.135)	0.783±0.012	0.895±0.002 (-0.016)	0.929±0.005	0.885±0.004 (-0.034)	0.912±0.004
Coauthor	Projection	0.847±0.009 (-0.106)	0.857±0.009	0.845±0.009 (-0.120)	0.849±0.009	0.728±0.011 (-0.228)	0.738±0.012
	Prediction	0.879±0.006 (-0.074)	0.889±0.006	0.932±0.004 (-0.033)	0.939±0.004	0.889±0.009 (-0.067)	0.903±0.009
	Embedding	0.888±0.007 (-0.065)	0.893±0.007	0.945±0.002 (-0.020)	0.950±0.003	0.885±0.031 (-0.071)	0.895±0.033
ACM	Projection	0.848±0.022 (-0.081)	0.848±0.021	0.871±0.014 (-0.064)	0.895±0.011	0.893±0.019 (-0.044)	0.922±0.019
	Prediction	0.840±0.023 (-0.089)	0.846±0.025	0.906±0.004 (-0.029)	0.929±0.007	0.893±0.010 (-0.044)	0.927±0.008
	Embedding	0.869±0.012 (-0.060)	0.878±0.009	0.902±0.011 (-0.033)	0.922±0.009	0.870±0.024 (-0.067)	0.896±0.024
Amazon	Projection	0.634±0.050 (-0.222)	0.623±0.069	0.780±0.034 (-0.173)	0.786±0.038	0.781±0.010 (-0.156)	0.792±0.011
	Prediction	0.804±0.030 (-0.052)	0.780±0.051	0.921±0.006 (-0.032)	0.937±0.006	0.893±0.013 (-0.044)	0.907±0.017
	Embedding	0.857±0.012 (0.001)	0.819±0.034	0.922±0.004 (-0.031)	0.938±0.004	0.906±0.020 (-0.031)	0.924±0.024

the inductive GNN models employed as surrogate models are powerful enough to extract information from the responses.

Number of Epochs and Batch Size. The number of epochs and batch size are the other two hyperparameters that may affect attack performance but can be controlled by the adversary. Respectively they control the number of complete passes through the training dataset and the number of samples processed before the GNN model is updated. Batch size may affect the speed and stability of the learning process, while the number of epochs may lead to overfitting. To this end, we fix the surrogate model to GraphSAGE and the target models to GIN and GAT to understand the impact of both hyperparameters. We use 150, 200, 250, 300 for the number of epochs, and 400, 600, 800, 1000 for bath sizes. The results are summarized in Figure 10 and Figure 11, respectively. Regarding the number of epochs, the accuracy is relatively stable with respect to different numbers of epochs (see Figure 10). Meanwhile, as we can observe in Figure 11, a larger batch size may have some negative impact on the attack accuracy.

Takeaways. Inductive GNN models employed as surrogate models are powerful enough to extract information from the responses. We observe that a large batch size may have a

TABLE X: The accuracy and fidelity score of Type II attacks using different response information on all the 6 datasets. Both average values and standard deviations are reported. The accuracy differences (in parenthesis) of the surrogate models to the target models are also reported. We use GIN as the surrogate model.

Dataset	$\mathcal{M}_S$ (GIN)	$\mathcal{M}_T$					
		GIN		GAT		SAGE	
		Accuracy	Fidelity	Accuracy	Fidelity	Accuracy	Fidelity
DBLP	Projection	0.715±0.028 (-0.157)	0.751±0.035	0.680±0.005 (-0.158)	0.701±0.006	0.724±0.025 (-0.134)	0.775±0.029
	Prediction	0.788±0.002 (-0.084)	0.839±0.005	0.775±0.003 (-0.063)	0.835±0.003	0.818±0.004 (-0.040)	0.899±0.006
	Embedding	0.785±0.012 (-0.087)	0.826±0.015	0.794±0.004 (-0.044)	0.846±0.015	0.803±0.004 (-0.055)	0.876±0.004
Pubmed	Projection	0.826±0.029 (-0.098)	0.851±0.034	0.736±0.022 (-0.169)	0.750±0.020	0.818±0.064 (-0.091)	0.855±0.076
	Prediction	0.873±0.004 (-0.051)	0.904±0.003	0.870±0.003 (-0.035)	0.898±0.002	0.876±0.002 (-0.033)	0.933±0.005
	Embedding	0.885±0.004 (-0.039)	0.913±0.004	0.882±0.001 (-0.023)	0.910±0.003	0.882±0.004 (-0.027)	0.933±0.007
Citeseer	Projection	0.665±0.026 (-0.246)	0.673±0.023	0.673±0.021 (-0.238)	0.679±0.014	0.726±0.030 (-0.193)	0.740±0.026
	Prediction	0.841±0.014 (-0.070)	0.877±0.010	0.880±0.008 (-0.031)	0.922±0.003	0.896±0.005 (-0.023)	0.952±0.006
	Embedding	0.814±0.014 (-0.097)	0.836±0.012	0.889±0.005 (-0.022)	0.924±0.004	0.884±0.004 (-0.035)	0.930±0.003
Coauthor	Projection	0.859±0.040 (-0.094)	0.849±0.040	0.839±0.053 (-0.126)	0.844±0.054	0.744±0.048 (-0.212)	0.751±0.048
	Prediction	0.951±0.001 (-0.002)	0.955±0.001	0.955±0.003 (-0.010)	0.960±0.002	0.956±0.001 (-0.000)	0.979±0.001
	Embedding	0.947±0.004 (-0.006)	0.953±0.003	0.951±0.001 (-0.014)	0.954±0.002	0.949±0.004 (-0.007)	0.963±0.006
ACM	Projection	0.856±0.036 (-0.073)	0.850±0.047	0.876±0.055 (-0.059)	0.886±0.056	0.893±0.030 (-0.044)	0.921±0.030
	Prediction	0.902±0.008 (-0.027)	0.899±0.012	0.928±0.005 (-0.007)	0.941±0.006	0.934±0.006 (-0.003)	0.962±0.004
	Embedding	0.901±0.011 (-0.028)	0.891±0.017	0.925±0.005 (-0.010)	0.937±0.004	0.928±0.007 (-0.009)	0.947±0.008
Amazon	Projection	0.653±0.052 (-0.203)	0.668±0.058	0.657±0.039 (-0.296)	0.656±0.041	0.764±0.029 (-0.173)	0.774±0.028
	Prediction	0.892±0.018 (0.036)	0.854±0.016	0.907±0.010 (-0.046)	0.914±0.005	0.943±0.005 (0.006)	0.953±0.006
	Embedding	0.912±0.010 (0.056)	0.860±0.006	0.933±0.005 (-0.020)	0.941±0.010	0.939±0.002 (0.002)	0.956±0.004

TABLE XI: The accuracy and fidelity score of Type II attacks using different response information on all the 6 datasets. Both average values and standard deviations are reported. The accuracy differences (in parenthesis) of the surrogate models to the target models are also reported. We use GAT as the surrogate model.

Dataset	$\mathcal{M}_S$ (GAT)	$\mathcal{M}_T$					
		GIN		GAT		SAGE	
		Accuracy	Fidelity	Accuracy	Fidelity	Accuracy	Fidelity
DBLP	Projection	0.702±0.023 (-0.170)	0.727±0.027	0.668±0.007 (-0.170)	0.693±0.011	0.746±0.023 (-0.112)	0.793±0.032
	Prediction	0.739±0.010 (-0.133)	0.784±0.012	0.779±0.005 (-0.059)	0.834±0.014	0.806±0.004 (-0.052)	0.881±0.005
	Embedding	0.741±0.006 (-0.131)	0.774±0.009	0.800±0.006 (-0.038)	0.853±0.008	0.743±0.032 (-0.115)	0.779±0.042
Pubmed	Projection	0.822±0.023 (-0.102)	0.850±0.024	0.746±0.054 (-0.159)	0.765±0.053	0.844±0.024 (-0.065)	0.889±0.030
	Prediction	0.837±0.008 (-0.087)	0.869±0.010	0.861±0.003 (-0.044)	0.896±0.004	0.861±0.002 (-0.048)	0.917±0.004
	Embedding	0.854±0.011 (-0.070)	0.880±0.012	0.873±0.004 (-0.032)	0.911±0.003	0.863±0.006 (-0.046)	0.909±0.010
Citeseer	Projection	0.704±0.017 (-0.207)	0.719±0.017	0.750±0.022 (-0.161)	0.762±0.025	0.768±0.019 (-0.151)	0.796±0.022
	Prediction	0.800±0.010 (-0.111)	0.838±0.014	0.883±0.005 (-0.028)	0.916±0.006	0.890±0.002 (-0.029)	0.941±0.003
	Embedding	0.807±0.023 (-0.104)	0.819±0.026	0.893±0.011 (-0.018)	0.932±0.007	0.876±0.005 (-0.043)	0.914±0.007
Coauthor	Projection	0.643±0.100 (-0.310)	0.649±0.101	0.762±0.077 (-0.203)	0.764±0.080	0.724±0.031 (-0.232)	0.733±0.031
	Prediction	0.823±0.052 (-0.130)	0.831±0.054	0.871±0.033 (-0.094)	0.877±0.034	0.795±0.029 (-0.161)	0.806±0.032
	Embedding	0.864±0.020 (-0.089)	0.871±0.021	0.882±0.036 (-0.083)	0.887±0.036	0.741±0.085 (-0.215)	0.750±0.087
ACM	Projection	0.834±0.011 (-0.095)	0.838±0.009	0.896±0.012 (-0.039)	0.921±0.011	0.910±0.008 (-0.027)	0.940±0.011
	Prediction	0.880±0.004 (-0.049)	0.889±0.021	0.915±0.007 (-0.020)	0.946±0.007	0.917±0.004 (-0.020)	0.956±0.004
	Embedding	0.909±0.006 (-0.020)	0.913±0.012	0.907±0.007 (-0.028)	0.928±0.008	0.908±0.015 (-0.029)	0.941±0.018
Amazon	Projection	0.674±0.040 (-0.182)	0.590±0.058	0.661±0.055 (-0.292)	0.557±0.060	0.739±0.015 (-0.198)	0.694±0.019
	Prediction	0.765±0.076 (-0.091)	0.653±0.074	0.881±0.024 (-0.072)	0.753±0.022	0.849±0.024 (-0.088)	0.752±0.029
	Embedding	0.853±0.035 (-0.003)	0.767±0.023	0.897±0.010 (-0.056)	0.869±0.014	0.832±0.043 (-0.105)	0.756±0.051

negative impact on the attack accuracy. Other hyperparameters such as hidden unit size and the number of epochs have a limited impact on the accuracy of the surrogate models.