Model Extraction Attacks on Graph Neural Networks: Taxonomy and Realization

Bang Wu
Monash University
bang.wu@monash.edu

Xiangwen Yang
Monash University
wayne.yang@monash.edu

Shirui Pan
Monash University
shirui.pan@monash.edu

Xingliang Yuan
Monash University
xingliang.yuan@monash.edu

ABSTRACT

Graph neural networks (GNNs) have been widely used to analyze the graph-structured data in various application domains, e.g., social networks, molecular biology, and anomaly detection. With great power, the GNN models, usually as valuable Intellectual Properties of their owners, also become attractive targets of the attacker. Recent studies show that machine learning models are facing a severe threat called Model Extraction Attacks, where a well-trained private model owned by a service provider can be stolen by the attacker pretending as a client. Unfortunately, existing works focus on the models trained on the Euclidean space, e.g., images and texts, while how to extract a GNN model that contains a graph structure and node features is yet to be explored.

In this paper, we explore and develop model extraction attacks against GNN models. Given only black-box access to a target GNN model, the attacker aims to reconstruct a duplicated one via several nodes he obtained (called *attacker nodes*). We first systematically formalise the threat modeling in the context of GNN model extraction and classify the adversarial threats into seven categories by considering different background knowledge of the attacker, e.g., attributes and/or neighbor connectives of the attacker nodes. Then we present the detailed methods which utilize the accessible knowledge in each threat to implement the attacks. By evaluating over three real-world datasets, our attacks are shown to extract duplicated models effectively, i.e., more than 89% inputs in the target domain have the same output predictions as the victim model.

1 INTRODUCTION

Graph data are ubiquitously used in many applications ,e.g., social media, document collections, and rating networks [14, 18, 20, 24–26, 42]. A graph with wealthy information often contains a group of attributed nodes and a set of edges. The nodes represent the instances with their features, while the edges represent the relationships among them. One of the prevalent tasks on processing the graph data is node classification. It aims at predicting the labels of the nodes based on their attributes, connectivity, and other connected nodes. For example, the type of users (nodes) in a social media can be inferred based on their personal profiles (attributes) and friendships (connectivity and other nodes). Among others, graph neural networks (GNNs), as graph-based machine learning models, have been widely deployed and offered state-of-the-art performance in the node classification tasks [10, 14, 25, 26, 35].

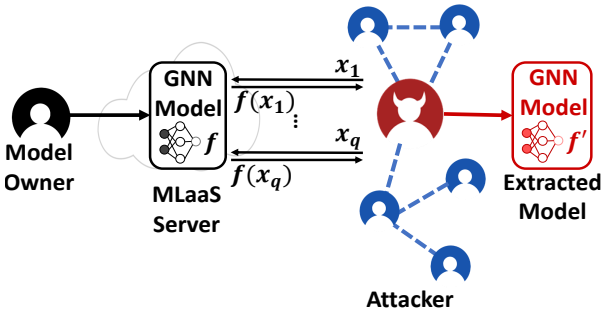


Figure 1: GNNs Model Extraction Attacks. A model owner provides a GNN model f and the service of prediction queries. An attacker extracts a surrogate model $f' \approx f$ based on the answers from the server.

While achieving high performance, the GNN models are also costly during the data gathering and training periods. For example, collecting biological data in a wet laboratory requires a large amount of resources and time [46]. Consequently, a well-trained GNN model is valuable and often considered as intellectual property of its owner. Catering the demands, lots of AI platforms, e.g., Amazon SageMaker and Google Cloud AutoML, provide privatization deployments for the model owners to sell their models with the license fee. Such commercialization draws much attention to the security of the models.

Previous researches have shown that attackers can steal several types of ML models by so-called Model Extraction Attack [32, 37]. They generate a sequence of queries to the target model and reconstruct a substitute model based on the input-output pairs of these queries. Since the input-output mappings contain sufficient information of the model prediction tasks, the extracted model can be very similar to the target model (e.g., generating the same predictions as the target model). However, existing attacks only target at the models with non-graph structures, e.g., MLP and CNN, while few studies focus on graph data. Thus, their threats to GNN models are still unclear.

In this paper, we explore and develop the model extraction attacks against GNNs. Specifically, referring a private model as the

target, an extraction attacker aims to construct a duplicated model with similar functionality via a sequence of queries from the nodes they obtained (called *attacker nodes*). Figure 1 illustrates an example of how our attack steals a GNN model on a social network. We consider an extraction attacker blending in with the normal users in the network. The attacker generates queries to the target model deployed in a machine learning as a service (MLaaS) and obtains the responses via its APIs. A extracted model can be eventually trained using these input-output pairs.

Unlike the attacks to other neural networks, extracting GNN models requires knowledge in addition to the mapping between the inputs and outputs. An attacker targeting at these graph based classifiers need to further consider the contributions of graph structures to classification tasks. Different from the models for non-graph structured data, a GNN model predict labels based on not only the input nodes but also their connectivity. For example, while the type of an image can be inferred by a CNN model individually, the prediction of a node label will consider all the attributes of itself and other connected nodes. Respectively, training of the GNNs requires both the training set of input-output pairs and the structure of the training graph. Therefore, GNN models contain the knowledge about the mapping between input nodes and output labels, as well as the interaction among connected nodes in the graphs. Such interaction related to the graph structure needs also to be extracted by the attacker to build a substitute model.

Developing model extraction attack on GNN models faces new challenge. The existing model extraction attacks do not consider the extra knowledge about graph structure, so their attack strategies cannot be directly applied to GNN models. The extracting attacker in GNNs should gather both the input-output query pairs and the graph structure to reconstruct a duplicated model. However, such knowledge may not be obtained by the attackers in real-world applications. For example, considering an online social network that contains both public and hidden user information, the attacker may have no access to the private data, such as the friendships (node connectives) or the personal information (node attributes) [15, 17]. Therefore, how to implement the extraction attacks with only missing or incomplete knowledge of the target model training graph is a non-trivial problem.

To address the above challenge and introduce effective model extraction attacks on GNNs, we first propose a comprehensive and formal framework of threat modeling by considering the attackers with different background knowledge. The knowledge includes three dimensions: the attributes of the attacker nodes, the partial graph consisted of the attacker nodes, an auxiliary sub-graph (shadow graph) including both its graph structure and attribute information. This sub-graph can be exclusive to the graphs that are used to train the target models but having similar attributes and graph structure. With or without this knowledge, we characterize our attacks into seven different types of model extraction attacks. For these seven attacks, we implement them in adaptive strategies based on their knowledge. Specifically, if the attacker knows graph structure but lacks node attributes knowledge, we design attribute synthesis algorithms to enrich the node attribute set and improve the attacks. If the obtained graph structure is corrupted, we utilize the known attributes to construct a surrogate graph by graph structure generation methods, i.e., Learning Discrete Structures [12]. The main contributions of our work are summarized as follows:

- We develop a series of GNN model extraction attacks that can steal a GNN model with a sequence of queries. The extracted model behaves similarly as the target victim model.
- We propose a framework of threat modeling in the context of GNNs, which formalizes and characterizes the attacker's knowledge from three dimensions: node attributes, graph structure and shadow sub-graph.
- We define seven types of extraction attacks with different adversarial knowledge and realize them via adaptive attack strategies. We implement them by utilizing known background knowledge and constructing a surrogate training graph to build a duplicated model.
- We evaluate our attacks over three real-world datasets. Our experiments show that, our attacks can effective extract a duplicated model similar as the target model. Most of the duplicated models achieve nearly equal accuracy as the target, and more than 85% prediction results from them are the same as the target.

The rest of the paper is organized as follows. Section 2 proposes the threat model. Section 3 introduces the detailed attack methodologies for our attacks. Section 5 shows our experimental results. Section 6 discusses some other attacks in GNNs and the attacks targeting at ML system's privacy. Section 7 provides the summary and conclusion of our paper.

2 PROBLEM FORMULATION AND BACKGROUND

In this section, we define the objective of our attacks. We first formalize the attack models and provide the formal definition of the proposed model extraction attacks. Then, we introduce the architecture of these target models which is prevalently used to evaluate the attacks in GNNs. Our attacks under these typical scenarios can also be extended to the models with other tasks.

2.1 Problem Definition

Definition 2.1. (Node Classification Model) Given an attributed graph $G = (V, E, X)$, a set of nodes V with node features X are connected by several edges E. A node classification model $f(.)$ can assign node labels Y to each node in the node set V corresponding to both their node features and the graph structure. We denote a classifier with parameters θ as $f_\theta(.)$. The classification result for this model on a node v_i (where $v_i \in V$) in the graph G is designated as $p_i = f_\theta(v_i)$. For a well-trained GNN model, p_i is expected to be the same as y_i (where y_i is the corresponding label of v_i).

As a target model, a node classifier $f_\theta(.)$ contains the knowledge which is the mapping from the nodes to their corresponding labels ($v_i \rightarrow y_i$). The model extraction attack aims at extracting this mapping and stealing the target private model without getting access to the model parameters θ. In general, considering an attack targeting at an ordinary deep neural network model $f(.)$, the attacker's goal is to extract the knowledge about the input-output mapping $x_i \rightarrow y_i$ [37]. The adversaries are assumed to have black-box access to the target model. Namely, they can generate a sequence

Notation	Explanation
G	An attributed graph of target model training
V	Node set of G
E	Edge set of G
X	Attribute set of V
Y	Label set of the nodes V
$f_\theta(.)$	A node classification model with parameters θ
P	Prediction result set of $f_\theta(v_i)$ for every node $v_i \in V$
G'	A shadow graph with the same domain as G
$V_{\mathcal{A}}$	Attacker node set
$V_{\mathcal{A},k-hop}$	k-hop neighbor node set of the attacker nodes $V_{\mathcal{A}}$
$E_{\mathcal{A}}$	Connectives among the attacker nodes $V_{\mathcal{A}}$
$E^*_{\mathcal{A}}$	Synthetic connectives among the attacker nodes $V_{\mathcal{A}}$
$E_{\mathcal{A},k-hop}$	k-hop neighbor connectives of the attacker nodes $V_{\mathcal{A}}$
$X_{\mathcal{A}}$	Attributes of the attacker nodes $V_{\mathcal{A}}$
$X^*_{\mathcal{A},k-hop}$	Synthetic attributes of the k-hop neighbours
D_i	Degree of the node v_i

Table 1: Notations and Explanations

of queries $(x_1, x_2, ..., x_q)$ to the target victim model and receive the labels of the predictions $(p_1, p_2, ..., p_q)$. Knowing only label is the most difficult setting for the attacker comparing with other works where the posteriors of the prediction are known [8, 33, 45]. Based on these query inputs and these response output labels, a model extraction attack can reconstruct a duplicated model by analyzing the relationship between them. When stealing a graph-structure model, we can extend this problem to GNNs as the following definition.

Definition 2.2. (Model Extraction Attacks in GNNs) A GNN model $f_\theta(.)$ is trained on an attributed graph $G = (V, E, X)$ for a classification task. An attacker with a black-box access to this model obtains a set of attacker nodes $V_{\mathcal{A}} \subset V$ which can be used to generate queries. The attacker can generate queries from any of these attacker nodes $v_a \in \{v_1, v_2, ..v_{\text{II}}\}$ and obtain their corresponding responses $f(v_a)$. The model extraction attack aims to reconstruct a surrogate model $f_{\theta'}(.)$ such that $\forall v_i \in V, f'_{\theta'}(v_i) = f_\theta(v_i)$, where V is a set for all the node in the graph and v_i is one of the node.

Considering a MLaaS system, a private model provided by an entity can be uploaded to the cloud server. This server provides query interface to the clients, while the clients can issue queries to the server and receive the responses. We consider an attacker with access to some of these clients, which is a common adversary in graph-based systems [9, 41, 47]. Namely they can generate queries and receive responses as an ordinary client. The model extraction attack aims at utilizing the information leaked from these input-output query pairs and extracting the knowledge about the private model. The notations we use throughout the paper are summarized in Table 1.

2.2 Background on Graph Convolution Network

In this paper, we consider a general graph convolution network (GCN) [25], indicated by $f_\theta(.)$, for the node classification task. In the convolutional layer of GCN, it explores the topological structure in spectral space and aggregates attribute information from the neighbor nodes followed by the non-linear transformation such as ReLU. The equation for a two-layer GCN is defined as:

$$f_\theta(A, X) = softmax(\hat{A} \cdot ReLU(\hat{A} \cdot X \cdot W^{(0)}) \cdot W^{(1)}), \quad (1)$$

where $\hat{A} = \hat{D}^{-1/2}\widetilde{A}\hat{D}^{-1/2}$ denotes the normalized adjacency matrix, $\widetilde{A} = A + I_N$ denotes adding the identity matrix I_N to the adjacent matrix A. $\hat{D}$ is the diagonal matrix with on-diagonal element as $\hat{D}_{ii} = \sum_j \widetilde{A}_{ij}$. $W^{(0)}$ and $W^{(1)}$ are the weights of first and second layer of GCN, respectively. $ReLU(0, a) = max(0, a)$ is adopted.

Considering the model extraction attacks against GCNs, the parameters of the targeted model are W=$\{W^{(0)}, W^{(1)}\}$. As a result, the goal of the attacks targeting at a GCN model becomes reconstructing the weights W.

Attack	X	A	G'	Attack	X	A	G'
−−	○	○	○	Attack-3	○	○	●
Attack-0	◐	◐	○	Attack-4	◐	◐	●
Attack-1	◐	○	○	Attack-5	◐	○	●
Attack-2	○	●	○	Attack-6	○	●	●

Table 2: Taxonomy of the proposed threat model. X represents the target dataset's nodes attributes, A represents the target dataset's graph structure, G' represents a shadow graph, and ●/◐/○means the attacker has complete/partial/no knowledge.

3 PROPOSED FRAMEWORK

3.1 Overview

Having introduced the objective of the model extraction attacks in GNNs, we will first present our proposed framework that categorizes the attacks. The extraction attacks happen when an attacker obtains some nodes in the target graph and then constructs the duplicated model. The common approach to implement the attack is to reconstruct the model by building the training data which consists of a sequence of input queries from the attacker and their corresponding output responses. These prediction responses can be utilized as the labels of the inputs, so the input-output pairs constitute the training data that we use to learn the functionality of the target models. Since the model represents the mapping between the input node features and the output labels, it is possible to learn a duplicated model based on these queries and responses.

Attack scenarios: We use a scenario of online social networks to demonstrate the practicality of our attacks. Here, an extraction attacker could be a party who aims to steal a private and valuable GNN model trained from an online social network system, such as Facebook or Twitter. The GNN model can predict the users (nodes) in the network based on their private but unlabeled profiles with different attributes, e.g., their ages and genders. To extract the model, the attacker will first attempt to control or recruit multiple users. Then, he will issue the queries to the target model from these users and capture the responses via the APIs of the service

providers. Besides, the attacker can gather extra information from social networks to facilitate the extraction. Some information is publicly available, such as the social structure, i.e., the connectives around a set of nodes. They are represented as the social relationships and can be captured by the web crawlers or other tools. Other information like users' profiles or the historical behavior records is the node attributes which help to predict their labels. They could be collected by hacking the corresponding databases or monitoring the actions of the target users. In practice, the adversaries may have different capabilities of collecting the extra knowledge. Based on various kinds of background knowledge, they can adaptively select the attack strategies to achieve better performance.

3.2 Attack Taxonomy

As discussed above, the targeted models are extracted based on the input-output pairs of queries. For the GNN models, the inputs are the node attributes, and the adjacency matrix derived from the graph structure. In practice, the attacker is assumed to only get access to a set of attacker nodes, i.e., a subset of the nodes in the entire graph. Therefore, they often do not have full knowledge of the inputs and outputs. In this paper, we propose different attack methods considering various background knowledge of the attacker. They are characterized by three dimensions as below:

Nodes' Attributes X of the Target Training Graph. This characterizes how much the attacker knows about the attributes X of the nodes V in the graph G used to train the target model. Generally, the attacker can have full access to the attacker nodes they obtain. Therefore, they can directly collect the node attributes for the applications that store them in each node, e.g., the users' profiles in a social network system are accessible for the end users. On the contrary, node attributes of other graph data with different classification tasks may not be obtained by the attacker. For example, the node attributes of the tasks predicting the nodes based on their purchased preferences may contain commercial information and are hidden to the attacker even they control these nodes. Note that, the attributes of other nodes who have not been compromised should be invisible to the attacker consistently [40, 41]. Therefore, we assume that the attacker cannot obtain the attribute knowledge except the attacker nodes.

Graph Structure A of the Target Training Graph. This characterizes how much the attacker knows the graph structure of the target graph G. Unlike the attributes which contain only the information from one node, the graph structure presents the relationship among multiple nodes. An attacker knowing the edges of the attacker nodes can construct a sub-graph consisted of both the attacker nodes and their neighbours. Besides, while the node attributes are considered as private data, the connectives such as friendships and following relations can be public. The attacker can reconstruct the target graph by crawling these public information [2, 4]. They can also utilize some graph structure reconstruction methods [11, 19] to obtain this knowledge.

Shadow Dataset $G' = (V', E', X')$. This represents a dataset in the same domain as the target dataset. An example could be a scenario when the target dataset and shadow dataset are from the same large network but different sub-graphs or communities [6, 13]. In practice, the model owner may only have the privilege or the capability to train their model based on the sub-graph in an extensive network. We assume the attacker may also have this privilege for another sub-graph as prior attack settings [15, 19, 36].

Combining above three dimensions, the knowledge of the attackers can be denoted as (X, A, G'). Considering whether the attacker has or has no knowledge of each item, we categorize the attacks into seven types summarized as Table 2. Notice that one scenario is not considered in our attacks, where the attacker knows nothing about the target models. Here, only the final prediction labels are leaked, so the attacker cannot gain any information about the link between the inputs and outputs. Namely, no model can be extracted under this assumption.

Algorithm 1 Algorithm for Attack-0

Input:
q attacker nodes' attributes $X_{\mathcal{A}} = \{x_{v_1}, x_{v_2}, ..., x_{v_q}\}$, q attacker nodes' query results $P_{\mathcal{A}} = \{p_{v_1}, p_{v_2}, ...p_{v_q}\}$, graph structure of the 2-hop neighbor nodes set of the attacker nodes $(V_{\mathcal{A},2-hop}, E_{\mathcal{A},2-hop})$, adjustment factor α.
Output:
Extracted Model $f_{\theta'}(.)$.

Generate adjacency matrix $A_{\mathcal{A},2-hop}$ for $(V_{\mathcal{A},2-hop}, E_{\mathcal{A},2-hop})$
Initialize a empty attribute set $X'_{\mathcal{A},2-hop}$
for $v_i \in V_{\mathcal{A},2-hop}$ **do**
  if $v_i \in V_{\mathcal{A}}$ **then**
    Collect x_{v_i} from $X_{\mathcal{A}}$
    Add x_{v_i} to $X'_{\mathcal{A},2-hop}$
    Label v_i as y'_{v_i} according to p_{v_i} in $P_{\mathcal{A}}$.
  else
    Initialize a empty attribute set $X'_{v_i,1-hop}$
    Initialize a empty attribute set $X'_{v_i,2-hop}$
    for $v_j \in V_{v_i,1-hop}$ **do**
      if v_j is in $V_{\mathcal{A}}$ **then**
        $x'_{v_j} = x_{v_j}/D_{v_j}$
        Add x'_{v_j}/D_{v_j} to $X'_{v_i,1-hop}$
    for $v_j \in V_{v_i,2-hop}$ and $\notin V_{v_i,1-hop}$ **do**
      if v_j is in $V_{\mathcal{A}}$ **then**
        $x'_{v_j} = x_{v_j}/D_{v_j}$
        Add x'_{v_j}/D_{v_j} to $X'_{v_i,2-hop}$
    $x'_{v_i} = \alpha \cdot \text{mean}(X'_{v_i,1-hop}) + (1-\alpha) \cdot \text{mean}(X'_{v_i,2-hop})$
    Add x'_{v_i} to $X'_{\mathcal{A},2-hop}$
Train 2-layer GCN $f_{\theta'}(.)$ based on $(X'_{\mathcal{A},2-hop}, A_{\mathcal{A},2-hop}, Y_{\mathcal{A}})$

4 ATTACK REALIZATION

In this section, we elaborate on the proposed attacks. Each of them is designed based on the corresponding adversarial knowledge.

4.1 Attack-0

We first consider a scenario where the attacker obtains a set of attacker nodes $V_{\mathcal{A}}$ and has both access to their attributes X and neighbor sub-graph structure A. These attacker nodes are randomly

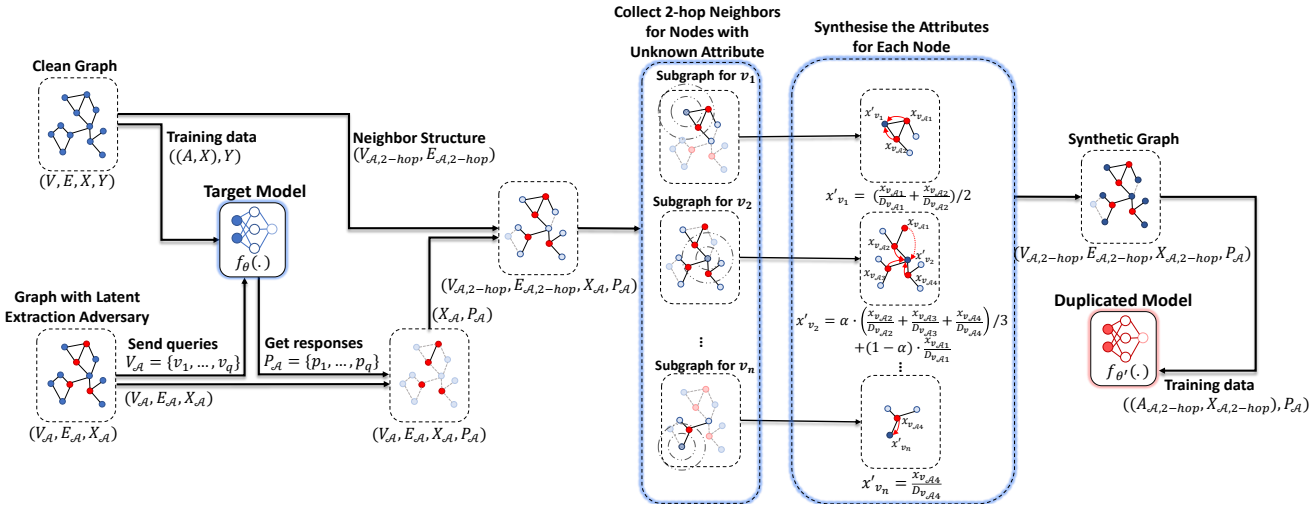


Figure 2: Illustration of Attack-0. After obtaining the query responses $P_{\mathcal{A}}$ from the target model f_θ, and the neighbor connectives $(V_{\mathcal{A},2-hop}, E_{\mathcal{A},2-hop})$ from the target graph, the attacker synthesizes the attributes $X'_{\mathcal{A},2-hop}$ for the neighbor nodes of the attacker nodes. The combination of the attacker nodes that have known attributes and labels, and the synthetic nodes that have generated attributes $(V_{\mathcal{A},2-hop}, E_{\mathcal{A},2-hop}, X'_{\mathcal{A},2-hop}, P_{\mathcal{A}})$ are used to train the duplicated GNN model $f_{\theta'}$ via semi-supervised learning.

chosen among the total node set V to imitate the real-world scenarios where every node in the victim graph can be a potential attacker node.

To extract the target models, the attacker intends to generate a training dataset for the duplicated model. We call it *attack graph* in the rest of our paper. The training data of the GNN model consists of node attributes, graph structure, and the node labels. The attacker proposes to obtain or generate the above items based on their adversarial knowledge. Specifically, the attack graph for the extracted model training can be built by three steps gathering each of above items. Figure 2 shows a procedure for procuring this attack graph.

Issuing queries and obtaining labels: In our assumptions, the attackers can obtain the attribute and query results of the attacker nodes. The results of the response queries from the attacker nodes can be considered as their node labels. Hence, they are utilized as the labeled nodes with known attributes to train a duplicated model for the node classification task.

Gathering neighbor connectives: Knowing input attributes, output predictions, and connectives among the attacker nodes, the attacker can naturally employ supervised learning to train the duplicated model. However, the graph structure of only the attacker nodes cannot reconstruct a satisfied model. For the prediction process of a GCN classifier, the features of each node will propagate along the edges among them and affect the neighbor nodes' features. In another words, the final prediction of a node is related to not only its own attributes but also the neighbours'.

Accordingly, the predictions of our attacker nodes are also affected by their neighbours. However, since our attacker nodes are randomly chosen from the entire graph, the attacker cannot guarantee that they are not solitary. Training the model by the attacker nodes isolated among the graph will desert the impacts from the neighbours of the attacker nodes and reduce the performance of attacks. Therefore, our design should rationally consider the attributes of the neighbours around these nodes. Specifically, the attacker will gather the connectives among the attacker nodes and their neighbours, which are considered as the graph structure of the attack graph.

Synthesising attributes for in-accessible nodes: In our assumptions, the attacker only knows the attributes and query results of the attacker nodes. Thus, the attacker proposes to synthesise the attributes for the neighbor nodes with unknown attributes. In practice, most nodes have similar attributes as their neighbours [5, 15, 30]. Based on this observation, the synthetic attributes can be the combination of their neighbor nodes' attributes. Formally, to synthesize the attributes of a target node x'_{v_i}, the attacker first gathers all its known-feature neighbours, including n 1-hop nodes $\{v_{1,1-hop}, ..., v_{n,1-hop}\}$ and m 2-hop nodes $\{v_{1,2-hop}, ..., v_{m,2-hop}\} \subset V_{\mathcal{A}}$. For each of them, the impact to the targets can be represented as $v_{j,k-hop}/D_j$, where D_j represents the degree of this neighbor node v_j. Considering an adjustment factor α to balance the effects from one or two hops nodes, the attacker synthesizes the feature of the target node as:

$$x'_{v_i} = \alpha \sum_{j=1}^{n} \frac{x_{v_j,1-hop}}{nD_j} + (1-\alpha) \sum_{j=1}^{m} \frac{x_{v_j,2-hop}}{mD_j} \tag{2}$$

Learning the extracted model: After generating the attributes for these nodes, the attacker can obtain a graph which includes all attacker nodes and their neighbours with the known or synthetic attributes. Based on our observations, most of the isolated attacker nodes can be connected after above processes. Therefore, this complementary structural graph can be the satisfied training data for our extracted model to learn the attribute propagation among the

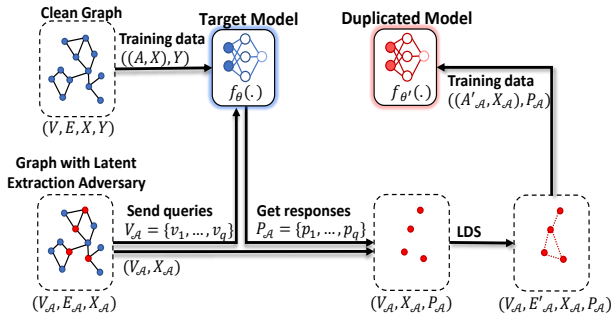


Figure 3: Illustration of Attack-1. After obtaining the query responses $P_{\mathcal{A}}$ from the target model f_θ, the attacker can only obtain discrete nodes with their attributes $X_{\mathcal{A}}$. A synthetic graph can be generated based on these attributes via graph generation method LDS [12]. Then the attacker nodes with attributes and labels and the synthetic graph structure $(V_{\mathcal{A}}, E'_{\mathcal{A}}, X_{\mathcal{A}}, P_{\mathcal{A}})$ are used to train the duplicated GNN model $f_{\theta'}$ via supervised learning.

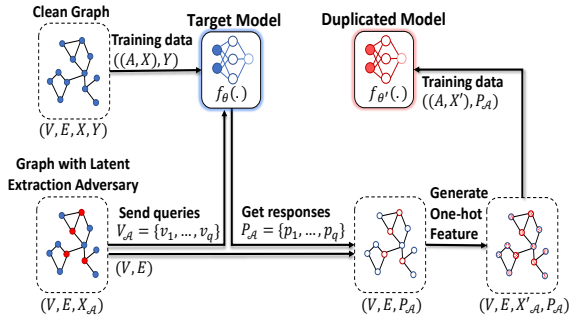


Figure 4: Illustration of Attack-2. After obtaining the query responses $P_{\mathcal{A}}$ from the target model f_θ, the attacker can only obtain the graph $(V, E, P_{\mathcal{A}})$ without node attributes. The attacker assigns one-hot vectors to every node as their synthetic attributes X' based on their graph index. Then the attacker nodes with attributes and labels and the synthetic graph structure $(V, E, X', P_{\mathcal{A}})$ are used to train the duplicated GNN model $f_{\theta'}$ via semi-supervised learning.

connectives. Based on this new graph with attributes for every node, labels for the attacker nodes, the attacker can train a node classification GNN model as the extracted model.

Noted that the attacker does not label the synthetic nodes. Unlike the labels of the attacker nodes that come from the query responses, the synthetic nodes are inaccessible to the attackers. He can neither modify the attributes of them nor send queries to the target models. As a result, only the attacker nodes can be labeled and the extracted models are trained via semi-supervised learning.

4.2 Attack-1

We then intensify the restriction to the attacker and consider the case when the attacker has only knowledge about the attributes of the attacker nodes $X_{\mathcal{A}}$. For this type of attack, the attacker also propose to first generate an attack graph for the extracted model training. A procedure of the attack is shown in Figure 3 and the attack graph is generated as follows.

Issuing queries and obtaining labels: Similar as Attack-0, the attributes and the query responses of the attacker nodes can be used as the labeled nodes in the attack graph.

Synthesising connectives among attacker nodes: Different from the Attack-0, the graph structure is unknown to the attacker. If all the attacker nodes are deemed to be isolated, the impacts from the neighbours of the attacker nodes cannot be taken into considerations. As discussed in Attack-0, it will affect the performance of our attacks. To solve this problem, the attacker proposes to infer the original graph based on the known attributes or construct a substitute graph.

Generally, the attributes of nodes in a graph and the connectives among them are tightly correlated [1, 39]. Thus, it is possible to infer or reconstruct the graph structure based on the node attributes. Based on this intuition, several prior studies about graph synthesis and generation have been developed to generate graphs [12, 29, 43]. Among others, we use a graph generation method called Learning Discrete Structures (LDS) [12] in our design. Unlike others, LDS generates the graphs by considering their performance on classification problems, which meets the tasks of our target models. This method is designed for training the GNN models with incomplete, corrupted or even unavailable graph. Its intuition is to generate the structure of the graph by approximating the discrete probability distribution on the edges via the nodes' attributes. The GCN models trained based on this strategy have been shown with even higher accuracy than the ordinary GCNs training. As a result, given the attributes of the attacker nodes, the attacker can synthesise the connectives among them and use the synthetic structure as the structure of the attack graph.

Learning the extracted model: After above steps, the attacker can obtain a substitute graph with attacker node attributes, corresponding prediction labels, and a generated graph structure. Then he can use supervised learning to train the duplicated models. Note that, due to the approximation of the edges distributions, the density of the generated graph can be set close to the target. Hence, most of the nodes generated via this method are not isolated and the attacker does not need to synthesise their neighbours as Attack-0.

4.3 Attack-2

We consider another scenario when the attacker obtains the entire graph structure knowledge A while having no access to any nodes in V even for the attacker nodes $V_{\mathcal{A}}$. Namely, they cannot obtain the node attributes X. In this attack, the attacker does not have any knowledge about the attributes. As a result, he proposes to build the attack graph by synthesising the attributes as Figure 4 shows. The detailed steps are:

Issuing queries and labeling the attacker nodes: Even though the attacker has no access to the node attributes, he can still obtain the responses and use them to label the attacker nodes. After generating attributes, these labeled nodes can by used during the extracted model training.

Gathering the target graph as attack graph: To reconstruct the target model, the attacker naturally utilizes the entire known graph structure to build the attack graph.

Assigning one-hot vectors as node attributes: Without any knowledge about the attributes, the attacker proposes to first synthesise them and build a surrogate training graph. As discussed in Section 2, the attribute of a node is also related to the entire graph structure and its position in it. To synthesise attributes associated to the structure knowledge, the attacker uses the index of the nodes to generate *one-hot vectors* as their attributes. For example, the attribute of v_1 will be $[1, 0, 0, ..., 0]$ while for v_2 is $[0, 1, 0, ..., 0]$. These attributes represent the identity of the nodes and contain information of the graph structure.

Note that, generating arbitrary features does not satisfy the training of the model with graph structure since they might bear no resemblance to their actual attributes. Meanwhile, it is also hard to reconstruct the original attributes of the target graph. As mentioned, the node attributes and the graph structure are tightly related. The objective of our attacks is to extract the mapping from node attributes to the node labels based on the graph structure. Thus, inferring the original attributes of the graph based on the graph structure and the node label or even only the node position can be considered as the reversed function of our target models. It is difficult to first learn this reversed mapping and then extract the target models.

Learning the extracted model: After synthesising the attributes, the attacker can then construct a surrogate model by these attributes, prediction labels of the attacker nodes, and the entire graph structure via semi-supervised learning. Different from previous attacks, the inputs of the surrogate models are the *one-hot vectors*. As a result, the nodes will be inferred via their indexes in the graph. Since the entire target graph structure is utilized, the extracted models can be used to classify all the nodes in the target graph as the target model.

4.4 Attack-3

We now consider the case when the attacker has knowledge about neither the node attributes nor their connectives. But we assume the attacker has access to a shadow graph $G' = (V', E', X')$ defined in Section 2. Under this adversarial assumption, the attacker has no knowledge about the target graph. Therefore, the extraction can only refer to the shadow graph. As introduced in Section 3, the shadow graph has the same domain as the target. Therefore, it is possible to utilize the knowledge from a shadow graph, i.e., using it as the attack graph.

Specifically, the attacker first gathers both the node attributes X' and the graph structure A' of a shadow graph with the same domain as the target. He can also obtain the corresponding labels Y' for some nodes in the shadow graph. Then, this shadow dataset $D' = (X', A', Y')$ can be used to train a surrogate model via semi-supervised learning. Since the dimensions of the node attributes from the graph with the same domain are also the same, the parameters of the surrogate model (the weights W) have the same size. Therefore, these weights can be used as the extracted model which achieves similar functionality as the target model if the target and shadow graphs are in the same domain.

4.5 Attack-4

In this attack, the attacker is assumed to have access to the attacker nodes $V_{\mathcal{A}}$ as Attack-0. Besides, he can collect a shadow graph G' as Attack-3. With both the background knowledge as Attack-0 and Attack-3, the attacker proposes to combine them together. In particular, an associated attack graph is built by combining the attack graphs for these two attacks.

The attacker first generates an attack graph consisting of attacker and synthetics nodes with the same strategy as Attack-0. Then, the shadow graph is set to be the second attack graph as Attack-3. Since the attacker nodes in the first attack graph are often not connected to the second attack graph, the attacker will not synthesise the connectives between them. It can avoid the the negative impacts among the attacker nodes and the shadow graph. Hence, the associated attack graph is consisted of these two isolated graph components. After that, the attacker can train the extracted model on this associated attack graph based on all the known node attributes, their graph structure, and the labeled nodes from both the shadow graph and the attacker nodes in the target graph.

4.6 Attack-5

This attack considers the case where the attacker has access to a shadow graph G' and also the attributes of the attacker nodes $X_{\mathcal{A}}$ in the target graph. The adversarial knowledge is from both Attack-1 and Attack-3. Similar as Attack-4, the attacker can combine them to utilize the background knowledge of $X_{\mathcal{A}}$ and G' in this attack.

To implement the attack, a graph generation method is used to construct a structural graph for the attacker nodes based on their attributes as Attack-1. The graph structure and the corresponding nodes with attributes and query responses consist of the first attack graph. The attacker again uses the shadow graph as the second attack graph. Due to the same reason as Attack-3, he will not synthesise connections between two attack graphs. Then, two attack graphs are associated to build the combined attack graph for training of the extracted model.

4.7 Attack-6

This attack considers the assumption that the attacker has no access to the node attributes but the entire graph structure knowledge A and a shadow graph G'. Compared with the Attack-2, the attacker can gather extra knowledge from the shadow graph. However, it is hard to directly utilize this node attribute knowledge and implement a similar design as Attack-2. In Attack-2, the attributes of the nodes are one-hot vectors corresponding to the node indexes. Thus, the dimension of the synthetic attributes will be the same as the node numbers which does not match to the original attributes.

To combine the Attack-2 and 3, the attacker can build the ensemble models. Two models utilizing different background knowledge via the methods as Attack-2 or Attack-3 are trained separately. Even though their inputs are different, both their outputs are the posteriors of the label of nodes. An attack model is trained to predict the final labels based on two posteriors. Specifically, the inputs of the attack model are the stack of the outputs from the two extracted models while the outputs are the final prediction labels. Since the attacker cannot obtain the posteriors from the target models of our

attacks, the attack models are first generated in the shadow graph. The detailed processes are:

Extracting models on shadow graph: We extract a model on the shadow graph via Attack-2 by only using its graph structure knowledge. The inputs of this model are one-hot vector attributes. Then, we extract another model on the shadow graph via Attack-3 by using its entire graph data.

Training an attack model: Since the attacker can obtain all knowledge about the node attributes, graph structure, he can feed them into the two models generated above and gather their output posteriors. Then the posteriors and their corresponding labels are used to train a simple MLP model to predict the final labels.

Extracting model on target graph: After building the attack model on the two emulated extracted models in shadow graph, the attacker can generate the real extracted models. To utilize the graph structure knowledge of the real target graph, the model is extracted via Attack-2.

Building the ensemble models: After training the two extracted models for Attack-2, Attack-3 and the attack model, the attacker can setup the ensemble model. The output posteriors of the two extracted model are fed into the attack model to generate the final predictions.

5 EXPERIMENTS

In this section, we present a comprehensive set of experiments to evaluate our attacks. We first introduce the experiment setting and then present the detailed results for each attack.

5.1 Experimental Setup

Datasets Three public datasets are used to evaluate our proposed attacks, including Cora, Citeseer, and Pubmed [25]. All of them are benchmark datasets which are widely used for the evaluations of node classification models. These three datasets are citation networks whose nodes represent the publications and edges are their citations. The detailed statistics of the datasets are shown in Table 4.

Datasets	Node Number	Edge Number
Cora	2708	5429
Citeseer	3327	4732
Pubmed	19717	44338

Table 4: Dataset Statistics

Datasets Configuration We configure the datasets for our different attacks. For Attack-0, Attack-1, and Attack-2 which do not contain the knowledge about the shadow dataset, we use the entire graph data to train the target models. For the Cora dataset, we split the network in 140 (about 5% of the total nodes) labeled nodes as training part, 300 (11%) labeled nodes as validation, and the rest of them unlabeled. Among the unlabeled nodes, we choose 1000 (37%) of them as the testing sets. For the Citeseer, we set 120 (4%) labeled nodes as training part, 500 (15%) labeled nodes as validation, and 1000 (30%) unlabeled nodes as the testing set. And for the Pubmed dataset, 60 (0.3%) labeled nodes are used as the training part, while 500 (2.5%) labeled nodes as validation. We again choose 1000 (5.1%) unlabeled nodes as the testing set.

For Attack-3, Attack-4, Attack-5, and Attack-6 using shadow dataset, we split the network into two parts: the graph for target model training and the graph assumed to be known by the attacker. To generate the shadow dataset, we first split the entire network into several communities by Clauset-Newman-Moore greedy modularity maximization [7], and then divide them into two datasets. For the Cora network, we generate the training graph for the target models which consists of 1408 (about 50%) nodes. The rest of them which consist of 1300 nodes are used as the shadow dataset. We split both the target and the shadow networks into training and testing parts. Other configurations for the datasets share the same settings as the datasets for Attack-0, Attack-1, and Attack-2.

Evaluation Metric We evaluate our attacks from two aspects based on the two different definitions about similar performance of extracting the models. The first one is the *fidelity* which evaluates how similar the surrogate models and the target models are. Specifically, it is defined as the percentage of the v_i in V where $f_{\theta'}(v_i) = f_\theta(v_i)$. It is calculated by dividing the number of the common predictions between two models with the number of the total testing inputs. For higher fidelity, the extracted models are expected to have more similar performance as the target models. Extracted models with high fidelity can be used when the attacker requires further analysis about the target models, e.g., being used in an adversarial attacks as a target with infinite queries. Another metric is the *accuracy* that represents how accurate the surrogate models are in testing data. Specifically, it is the percentage of the v_i in V where $f_{\theta'}(v_i) = y_{v_i}$. It is calculated by dividing the number of the correct classified nodes with the number of the total testing nodes. In this case, the attacker aims to extract the models suitable for the target application tasks.

Models In our experiments, we consider the case where the target model is a 2-layer graph convolution network, introduced in Eq. 1. The number of features in the hidden layer is 16. The activation function for the hidden layer is ReLU and for the output layer is softmax. We also apply a dropout layer with 0.5 dropout rate after the hidden layer. We use the Adam optimizer with a learning rate of 0.02 and training epochs of 200. The loss function of our model is negative log likelihood loss.

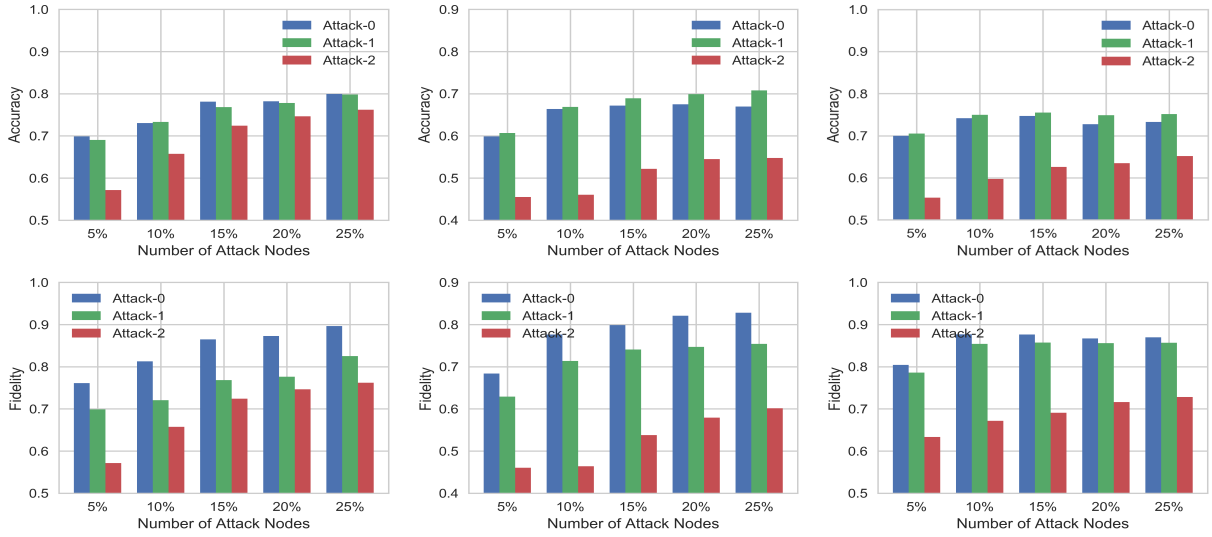
5.2 Attack Performance

Overview: Table 3 shows an overview of the performance of our seven attacks. For Attack-0, Attack-1 and Attack-2, the numbers of the attacker nodes obtained by the attacker are chosen to be about 25% of the total nodes in the target networks. For Attack-3, Attack-4, Attack-5, and Attack-6, the size of the shadow graph is set to be almost the same as the target, and the attacker is assumed to obtain fewer nodes which is about 10%. It can be found that our attacks achieve nearly equal accuracy as the target model as the baseline accuracy. Meanwhile, most of our attacks gain about 80% fidelity, which means that our extracted models mostly predict the inputs as the targets. We highlight the attacks with best performance among others. Detailed discussions for each attack are presented as follow.

Attack-0 Attack-0 is shown to achieve the highest performance in fidelity since their training data is the most similar as the target

Metrics	Accuracy			Fidelity		
Dataset	Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed
Target Model	0.816	0.713	0.800	–	–	–
Attack-0	**0.799 ± 0.009**	0.684 ± 0.016	0.736 ± 0.004	**0.896 ± 0.008**	**0.848 ± 0.019**	**0.890 ± 0.007**
Attack-1	0.798 ± 0.006	**0.708 ± 0.007**	**0.751 ± 0.003**	0.825 ± 0.007	0.754 ± 0.005	0.857 ± 0.003
Attack-2	0.762 ± 0.012	0.548 ± 0.004	0.652 ± 0.036	0.809 ± 0.006	0.602 ± 0.003	0.728 ± 0.035
Target Model	0.816	0.697	0.806	–	–	–
Attack-3	0.809 ± 0.007	0.692 ± 0.004	0.799 ± 0.001	0.790 ± 0.005	0.714 ± 0.002	0.818 ± 0.009
Attack-4	0.801 ± 0.009	**0.708 ± 0.002**	**0.800 ± 0.008**	0.790 ± 0.011	**0.736 ± 0.008**	**0.837 ± 0.004**
Attack-5	**0.832 ± 0.004**	0.699 ± 0.001	0.799 ± 0.002	**0.807 ± 0.002**	0.727 ± 0.002	0.818 ± 0.003
Attack-6	0.800 ± 0.017	0.6490 ± 0.017	0.737 ± 0.092	0.791 ± 0.019	0.7312 ± 0.019	0.813 ± 0.086

Table 3: Model accuracy/fidelity for all attack on three different datasets. Attack-0, Attack-1, and Attack-2 target at the model trained in the entire dataset. Attack-3, Attack-4, Attack-5, and Attack-6 target at the model trained in a sub-graph split from the entire graph. Best results are highlighted in bold.



(a) Cora (b) Citeseer (c) Pubmed

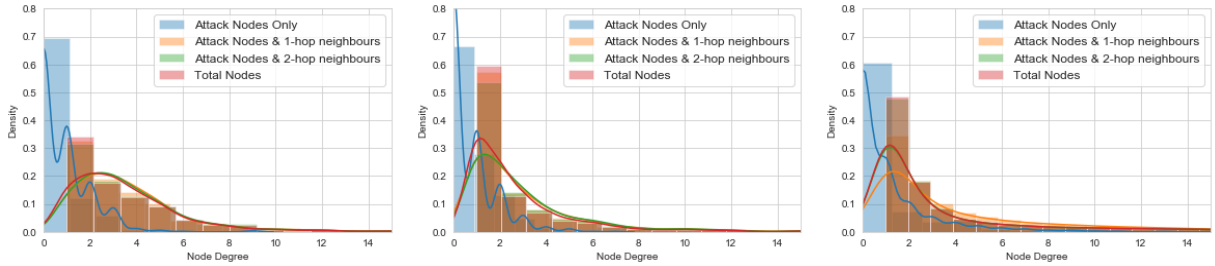
Figure 5: Impact of the number of the attacker nodes in Attack-0, Attack-1, and Attack-2

Metric	Dataset	Without Synthetic	First Order Neighbor Synthetic	Second Order Neighbor Synthetic
Accuracy	Cora	0.797 ± 0.012	0.799 ± 0.009	0.793 ± 0.008
	Citeseer	0.688 ± 0.013	0.684 ± 0.016	0.681 ± 0.017
	Pubmed	0.735 ± 0.027	0.736 ± 0.004	0.731 ± 0.013
Fidelity	Cora	0.869 ± 0.012	0.896 ± 0.008	0.889 ± 0.009
	Citeseer	0.816 ± 0.030	0.848 ± 0.019	0.834 ± 0.015
	Pubmed	0.879 ± 0.030	0.890 ± 0.007	0.886 ± 0.015

Table 5: Impact of the synthetic nodes for Attack-0.

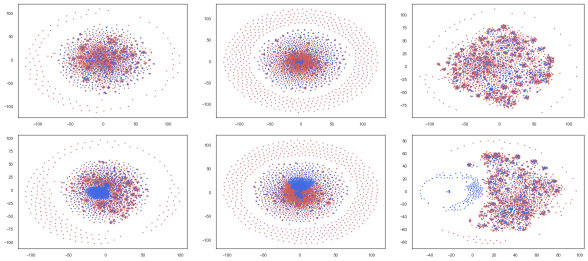
model. We analyze their performance by adjusting several factors in the design.

Figure 7 shows the relationship among the number of the attacker nodes and the fidelity/accuracy of the surrogate models from 5% of the total nodes to 25%. For larger numbers of attacker nodes, both accuracy and fidelity increase. The accuracy of the extracted model achieves about 79.9% in the Cora dataset which is very closed to the target model 81.5%. And the fidelity of the duplicated model is about 90%. For the Citeseer, the accuracy increases from 59.9% to 67.0% when obtaining the attacker nodes from 5% to 25%. The accuracy with about 25% nodes is close to the baseline accuracy 70.0%. The fidelity reaches 82.8% when the number of the attacker node is about 25% of the nodes in total graph. When attacking the model trained in Pubmed, the accuracy of the duplicated model increases from 70% to 73%.



(a) Cora (b) Citeseer (c) Pubmed

Figure 6: Degree Distribution for Attack-0



(a) Cora (b) Citeseer (c) Pubmed

Figure 7: Feature distribution of the nodes including only first order synthetic neighbor nodes (Upper) and both first and second order (Lower) for Attack-0 projected into a 2-dimension space using t-SNE.

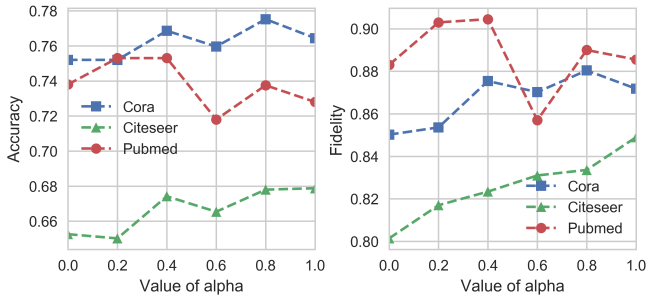


Figure 8: Impact of the adjustment factor α for Attack-0

We also evaluate how synthesising the neighbours affects our attack performance. Table 5 shows the accuracy and fidelity of the attack with and without the synthetic nodes. It can be found that, synthesising the attributes for the neighbours of the attacker nodes can improves the fidelity of our attacks. We also evaluate the attack performance when synthesising more neighbor nodes. It is shown that too many synthetic nodes will hurt our attacks. We compare the degree distribution and the feature distribution of the graph generating by different strategies in Figure 6 and 7. It is shown that the graph generated by synthesising only the first order achieves the most similar distribution as the target graph that matches to our attack results.

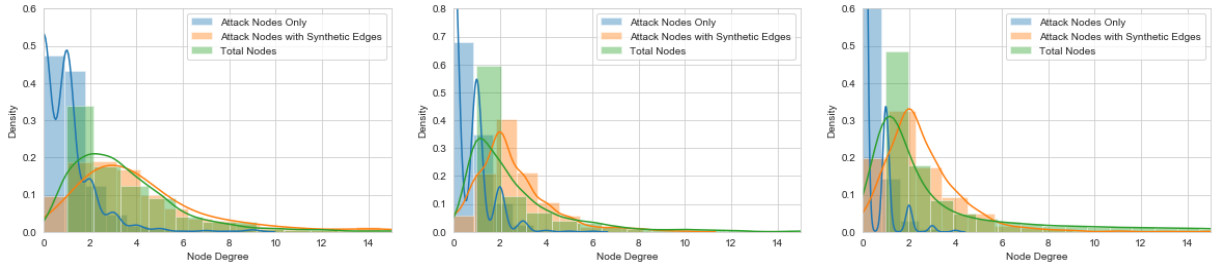
We now discuss the impact of the adjustment factor α. Figure 8 shows both the accuracy and fidelity of the attack with variant α. The experiments show that this factor does affect the attack performance but mostly inside ±5%. We can also find that the attack performance raises when α increases for both Cora and Citeseer. Larger α means the synthetic attributes of the nodes are more based on their 1-hop neighbours. This is reasonable since the relationship between the synthetic nodes to their 1-hop neighbours is stronger than the 2-hop neighbours. Meanwhile, the performance from the Pubmed is undulate. To achieve the best attack performance, the attacker can carefully choose the adjustment factor by considering the characteristic of the graph.

Attack-1 In Attack-1, only the attributes of the attacker nodes are known to the attackers while their connectivities are unknown. Therefore, we generate the graph structure based on these node features. Figure 5 shows the relationship between the number of attacker nodes and the attack performance. Similar as Attack-0, more attacker nodes can significantly increase the accuracy and fidelity of the extracted models.

To clearly show how the graph structure generation contributes to our design, we set a baseline, which uses deep neural networks to infer the output labels only based on the input features. The results for both accuracy and fidelity of the surrogate models in three different dataset are shown as Table 6. In Cora dataset, our design achieves about 79.8 for the accuracy of the extracted model while DNNs have only 57.7%. And our attack improves the fidelity from 59.0% to 82.5%.

To show how the generated graph structure matches to the original graph, we also evaluate the degree distribution of the graph generated by our attack methods. Notice that the attribute distribution should be very similar to the target graph since the attributes in Attack-1 are all from the attacker nodes which is considered as sample of the original. Figure 9 shows the comparison between the degree distribution of the generated graph and the target graph. It can be found that they are more similar comparing the distribution without graph generation. This demonstrates that using graph structure generation is a good approach to help reconstruct the graph structure and further improve our attack.

Attack-2 This attack considers the scenario when the attacker has only knowledge about the graph structure. The results for both accuracy and fidelity are shown in Table 3. And Figure 5 shows how the number of the attacker nodes affects our attack. Similarly,



(a) Cora (b) Citeseer (c) Pubmed

Figure 9: Degree Distribution for Attack-1

Metric	Dataset	Simple DNN	LDS-GNN [12]
Accuracy	Cora	0.577 ± 0.004	0.798 ± 0.006
	Citeseer	0.596 ± 0.004	0.708 ± 0.007
	Pubmed	0.727 ± 0.006	0.751 ± 0.003
Fidelity	Cora	0.590 ± 0.010	0.825 ± 0.007
	Citeseer	0.632 ± 0.005	0.754 ± 0.005
	Pubmed	0.761 ± 0.005	0.857 ± 0.003

Table 6: Fidelity/accuracy for Attack-1

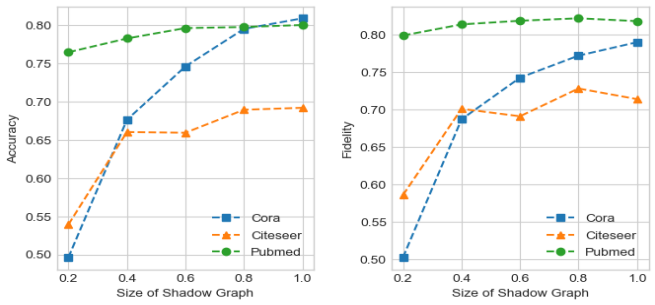


Figure 10: Impact of the shadow graph size for Attack-3

both accuracy and fidelity increase when obtaining more attacker nodes.

Notice that, Attack-2 has the worst performance comparing with Attack-0 and Attack-1. This might cause by the less similarity of the synthetic items for the attack graph we generated. For other attacks, our design can generate node attributes or connectives similar as the target graph. However, due to the lack of knowledge about the attributes, the synthetic attributes of the attack graph for this type of attacks are one-hot vectors which are far from the target. If the attacker can obtain some knowledge about the node attributes which can be used to synthetic similar attributes, the performance can be improved.

Attack-3 For Attack-3, we consider the case where the attacker can only have a shadow graph defined in Section 2 without knowing both attributes and graph structures for the target models. The results for both accuracy and fidelity are shown in Table 3. Compared to previous attacks with some background knowledge of the target graph, the accuracy of this type of attack is similar or even better. Thus, obtaining the complete graph data can achieve high accuracy if the shadow graph has the same domain as the target. However, the fidelity of the attack is significantly smaller than previous attacks since the training graph data of the target model is entirely different from our extracted graph. Our target model is build as the GCN model which is transductive, so it is hard to gain an extracted model with similar functionality.

We also analyze the effect for different knowledge of the shadow sub-graph. Figure 10 shows the relationship between the attack performance and the size of shadow graph. The x-axis represents the ratio of the size of the shadow graph to the target graph. It can be found that while knowing larger size of the shadow graph, the accuracy of the surrogate models increases a lot. It is obvious since the attacker can extract more knowledge from a larger training graph. It can also be found that the fidelity of the surrogate models becomes saturated even the shadow graph size becomes larger. As discussed, the target GCN model is transductive, which makes the attacker difficult to obtain an highly equivalent model. Therefore, the fidelity of our attack will reach the ceiling when the size of the shadow graph keeps increasing.

Attack-4 Now we consider the case when the the attacker has access to a shadow graph as well as some attacker nodes in the target graph. Based on Table 3, it can be found that Attack-4 achieves higher accuracy and fidelity than Attack-3. It demonstrates that obtaining extra knowledge can lead to better attack performance. Note that, the improvement of the fidelity is more significant than the accuracy especially for the Citeseer and Pubmed dataset. This shows that the knowledge from the target graph can be helpful to extract models with similar prediction as the target model.

Attack-5 Now we consider the case when the attacker has access to a shadow graph and also some attacker nodes in the target graph. Based on Table 3, it can be found that this attack achieves slightly higher accuracy and fidelity than the Attack-3 and nearly equal attack performance as the Attack-4. But since the knowledge of Attack-5 is less than Attack-4 (the neighbor graph structure is unknown), the fidelity is lower which is similar as the comparison between Attack-0 and 1. It demonstrates that obtaining more background knowledge can enhance our extraction attacks more.

Attack-6 Finally we discuss the attack when the attacker has both knowledge about the graph structure and the shadow graph. With the help of shadow graph, the overall performance of Attack-6 is significantly higher Attack-2. It demonstrates that introducing knowledge about the node attributes can improve the attack performance. It also achieves comparative performance as the Attack-4 and Attack-5.

6 RELATED WORKS

Model Extraction Attacks: Model extraction attacks targeting at the confidentiality of ML systems have become paramount and been explored in lots of studies [20]. Tramèr *et al.* [37] propose the first model extraction attacks against the linear ML models via Prediction APIs. They reconstruct the model by solving the equations built by the queries, and the labels or confidence values. They also use a path finding approach to attack the decision tree models. Chandrasekaran *et al.* [3] use the advancements in the active learning domain to implement the model extraction attacks that also target at linear models. Later, more studies consider attacking complex ML models, e.g., Neural Networks. Milli *et al.* [31] provide an gradient-based algorithm which extracts a two-layer ReLU network by carefully choosing the query inputs. Pal *et al.* [34] demonstrate an attack on DNNs for both image and text classification tasks with active learning strategies. Silva *et al.* [8] create a copycat CNN by simply querying a target black-box CNN using ordinary natural and non problem related image inputs. Orekondy *et al.* [33] propose the attack by training a "knockoff" model which aims to match or even exceed the accuracy of the target model by generating query-prediction pairs.

Several approaches have also been proposed to defend against model extraction attacks but they are not suitable in our attacks. Some of them propose to hide or add noise to the output probabilities while maintaining the label outputs [3, 27, 37]. But they are less effective facing the label-based extraction attacks like our design. Others try to monitor each query and differentiate the adversarial ones by analyzing the input distribution or the output entropy [22, 23]. However, they do not consider the graph structure and are not optimized for GNN models.

Attacks on Graph Neural Networks: Many studies have explored the vulnerability in GNNs. Most of them are adversarial attacks which target at the integrity of the GNN systems. Zügner *et al.* [47] propose a scalable greedy approximation scheme to find the perturbation attacking the node classification GNNs. They evaluate both node attribute and graph structure perturbation and compare their effectiveness. Wu *et al.* [41] generate the adversarial inputs by introducing integrated gradients which clearly shows the effect of perturbing certain features or edges. Zhang *et al.* [44] propose a collection of data poisoning attack strategies, by manipulating the facts on the target graph. Li *et al.* [28] study the attack on the graph learning based community detection models via hiding a set of nodes based on their surrogate model.

Recent studies also draw the attentions of attacking the confidentiality of GNNs. A set of advanced attacks called membership inference attacks aim to infer whether a data sample has been used during the target model training [16, 21, 38]. Besides, He *et al.* [19] apply link stealing attacks against GNNs which can infer whether there is a link between two nodes on their training graph.

7 CONCLUSION AND FUTURE DIRECTIONS

In this paper, we demonstrate a model extraction attack against GNNs. We first generates legitimate-looking queries as the normal nodes among the target graph, then utilize the query responses and accessible structure knowledge to reconstruct the model. We characterize the problem into seven threat models considering different knowledge of the attacker. Then we accordingly propose seven attacks based on the knowledge and the query responses. Our experimental results show that our attack obtains surrogate models with similar predictions as the targets.

An interesting research direction is the defence against the extraction attacks in GNNs. Existing methods like monitoring or filtering the input queries [22, 23] might be extended and combined with structure analysis for implementing effective defense strategies. We leave it as future work.

REFERENCES

[1] Deepak Bhaskar Acharya and Huaming Zhang. [n.d.]. Feature Selection and Extraction for Graph Neural Networks. In *Proc. ACM SE '2020.*

[2] Salvatore Catanese, Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Alessandro Provetti. [n.d.]. Crawling Facebook for social network analysis purposes. In *Proc. WIMS 2011.*

[3] Varun Chandrasekaran, Kamalika Chaudhuri, Irene Giacomelli, Somesh Jha, and Songbai Yan. 2018. Exploring connections between active learning and model extraction. *arXiv preprint arXiv:1811.02054* (2018).

[4] Duen Horng Chau, Shashank Pandit, Samuel Wang, and Christos Faloutsos. [n.d.]. Parallel crawling for online social networks. In *Proc. WWW 2007.*

[5] Jinyin Chen, Xiang Lin, Ziqiang Shi, and Yi Liu. 2020. Link Prediction Adversarial Attack Via Iterative Gradient Attack. *IEEE Trans. Comput. Soc. Syst.* 7 (2020).

[6] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. [n.d.]. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *Proc. KDD 2019.*

[7] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical review E* (2004).

[8] Jacson Rodrigues Correia da Silva, Rodrigo Ferreira Berriel, Claudine Badue, Alberto Ferreira de Souza, and Thiago Oliveira-Santos. [n.d.]. Copycat CNN: Stealing Knowledge by Persuading Confession with Random Non-Labeled Data. In *Proc. IJCNN 2018.*

[9] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. [n.d.]. Adversarial Attack on Graph Structured Data. In *Proc. ICML 2018.*

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. [n.d.]. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Proc. NeurIPS 2016.*

[11] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. 2020. Quantifying Privacy Leakage in Graph Embedding. (2020). arXiv:2010.00906

[12] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. [n.d.]. Learning Discrete Structures for Graph Neural Networks. In *Proc. ICML 2019 (Proceedings of Machine Learning Research).*

[13] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. [n.d.]. Large-Scale Learnable Graph Convolutional Networks. In *Proc. KDD 2018.*

[14] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. [n.d.]. Neural Message Passing for Quantum Chemistry. In *Proc. ICML 2017.*

[15] Neil Zhenqiang Gong and Bin Liu. [n.d.]. You Are Who You Know and How You Behave: Attribute Inference Attacks via Users' Social Friends and Behaviors. In *Proc. USENIX Security 16.*

[16] Neil Zhenqiang Gong and Bin Liu. 2018. Attribute Inference Attacks in Online Social Networks. *ACM Trans. Priv. Secur.* 21 (2018).

[17] Payas Gupta, Swapna Gottipati, Jing Jiang, and Debin Gao. [n.d.]. Your love is public now: questioning the use of personal information in authentication. In *Proc. ASIA CCS '13.*

[18] William L. Hamilton, Zhitao Ying, and Jure Leskovec. [n.d.]. Inductive Representation Learning on Large Graphs. In *Proc. NeurIPS 2017.*

[19] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. 2020. Stealing Links from Graph Neural Networks. (2020). arXiv:2005.02131

[20] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. [n.d.]. High Accuracy and High Fidelity Extraction of Neural Networks. In *Proc. {USENIX} Security 2020.*

[21] Jinyuan Jia and Neil Zhenqiang Gong. [n.d.]. AttriGuard: A Practical Defense Against Attribute Inference Attacks via Adversarial Machine Learning. In *Proc. USENIX Security 2018.*

[22] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N. Asokan. [n.d.]. PRADA: Protecting Against DNN Model Stealing Attacks. In *Proc. EuroS&P 2019.*

[23] Manish Kesarwani, Bhaskar Mukhoty, Vijay Arya, and Sameep Mehta. [n.d.]. Model Extraction Warning in MLaaS Paradigm. In *Proc. ACSAC 2018.*

[24] Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. [n.d.]. Learning Combinatorial Optimization Algorithms over Graphs. In *Proc. NeurIPS 2017.*

[25] Thomas N. Kipf and Max Welling. [n.d.]. Semi-Supervised Classification with Graph Convolutional Networks. In *Proc. ICLR 2017.*

[26] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. [n.d.]. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *Proc. ICLR 2019*.

[27] Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. [n.d.]. Defending Against Neural Network Model Stealing Attacks Using Deceptive Perturbations. In *2019 IEEE Security and Privacy Workshops*.

[28] Jia Li, Honglei Zhang, Zhichao Han, Yu Rong, Hong Cheng, and Junzhou Huang. [n.d.]. Adversarial Attack on Community Detection by Hiding Individuals. In *Proc. WWW 2020*.

[29] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter W. Battaglia. 2018. Learning Deep Generative Models of Graphs. *CoRR* abs/1803.03324 (2018).

[30] Peiyuan Liao, Han Zhao, Keyulu Xu, Tommi S. Jaakkola, Geoffrey J. Gordon, Stefanie Jegelka, and Ruslan Salakhutdinov. 2020. Graph Adversarial Networks: Protecting Information against Adversarial Attacks. *CoRR* abs/2009.13504 (2020).

[31] Smitha Milli, Ludwig Schmidt, Anca D. Dragan, and Moritz Hardt. 2019. Model Reconstruction from Model Explanations. In *Proc. the Conference on Fairness, Accountability, and Transparency*.

[32] Seong Joon Oh, Bernt Schiele, and Mario Fritz. 2019. Towards Reverse-Engineering Black-Box Neural Networks. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*.

[33] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. [n.d.]. Knockoff Nets: Stealing Functionality of Black-Box Models. In *Proc. CVPR 2019*.

[34] Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish K. Shevade, and Vinod Ganapathy. 2019. A framework for the extraction of Deep Neural Networks by leveraging public data. *CoRR* abs/1905.09165 (2019).

[35] Trang Pham, Truyen Tran, Dinh Q. Phung, and Svetha Venkatesh. 2017. Column Networks for Collective Classification. In *Proc. AAAI*.

[36] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. [n.d.]. Membership Inference Attacks Against Machine Learning Models. In *Proc. SP 2017*.

[37] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. [n.d.]. Stealing Machine Learning Models via Prediction APIs. In *Proc. USENIX Security 16*.

[38] B. S. Vidyalakshmi, Raymond K. Wong, and Chi-Hung Chi. [n.d.]. User Attribute Inference in Directed Social Networks as a Service. In *Proc. SCC 2016*.

[39] Binghui Wang, Tianxiang Zhou, Minhua Lin, Pan Zhou, Ang Li, Meng Pang, Cai Fu, Hai Li, and Yiran Chen. 2020. Evasion Attacks to Graph Neural Networks via Influence Function. *CoRR* abs/2009.00203 (2020).

[40] Xiao Wang, Di Jin, Xiaochun Cao, Liang Yang, and Weixiong Zhang. 2016. Semantic Community Identification in Large Attribute Networks. In *Proc. AAAI*.

[41] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. [n.d.]. Adversarial Examples for Graph Data: Deep Insights into Attack and Defense. In *Proc. IJCAI 2019*.

[42] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. [n.d.]. Session-Based Recommendation with Graph Neural Networks. In *Proc. AAAI 2019*.

[43] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. GraphRNN: A Deep Generative Model for Graphs. *CoRR* abs/1802.08773 (2018).

[44] Hengtong Zhang, Tianhang Zheng, Jing Gao, Chenglin Miao, Lu Su, Yaliang Li, and Kui Ren. [n.d.]. Data Poisoning Attack against Knowledge Graph Embedding. In *Proc. IJCAI 2019*.

[45] Huadi Zheng, Qingqing Ye, Haibo Hu, Chengfang Fang, and Jie Shi. [n.d.]. BDPL: A Boundary Differentially Private Layer Against Machine Learning Model Extraction Attacks. In *Proc. ESORICS 2019 (Lecture Notes in Computer Science)*.

[46] Marinka Zitnik, Jure Leskovec, et al. 2018. Prioritizing network communities. *Nature communications* (2018).

[47] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. [n.d.]. Adversarial Attacks on Neural Networks for Graph Data. In *Proc. IJCAI 2019*.