Revisiting Black-box Ownership Verification for Graph Neural Networks

Ruikai Zhou*, Kang Yang*, Xiuling Wang†, Wendy Hui Wang†, Jun Xu*
*University of Utah, †Stevens Insititute of Technology

***Abstract*—Graph Neural Networks (GNNs) have emerged as powerful tools for processing graph-structured data, enabling applications in various domains. Yet, GNNs are vulnerable to model extraction attacks, imposing risks to intellectual property. To mitigate model extraction attacks, model ownership verification is considered an effective method. However, throughout a series of empirical studies, we found that the existing GNN ownership verification methods either mandate unrealistic conditions or present unsatisfactory accuracy under the most practical settings—the black-box setting where the verifier only requires access to the final output (e.g., posterior probability) of the target model and the suspect model.**

Inspired by the studies, we propose a new, black-box GNN ownership verification method that involves local independent models and shadow surrogate models to train a classifier for performing ownership verification. Our method boosts the verification accuracy by exploiting two insights: (1) We consider the overall behaviors of the target model for decision-making, better utilizing its holistic fingerprinting; (2) We enrich the fingerprinting of the target model by masking a subset of features of its training data, injecting extra information to facilitate ownership verification.

To assess the effectiveness of our proposed method, we perform an intensive series of evaluations with 5 popular datasets, 5 mainstream GNN architectures, and 16 different settings. Our method achieves nearly perfect accuracy with a marginal impact on the target model in all cases, significantly outperforming the existing methods and enlarging their practicality. We also demonstrate that our method maintains robustness against adversarial attempts to evade the verification.

1. Introduction

Graph Neural Networks (GNNs) [15], [22], [48], [55], [58] are a class of neural networks designed to operate on graph data. GNNs leverage the inherent structure of graphs to learn complex patterns in data, supporting downstream tasks like node classification [11], [15], [22], [26], [48], [58] and graph classification [7], [23], [62]. GNNs have gained popularity and enabled applications in various domains, such as social network analysis [8], [50], recommendation systems [9], [56], cybersecurity [35], [66], etc.

Not surprisingly, the significance of GNNs has attracted many attacks. A major family against GNNs is *model extraction attacks* [12], [13], [42] where the adversaries extract information through query access to a *target model* and train a *surrogate model* to approximate the target model. Commonly, model extraction attacks against GNNs work in a black-box manner [6], [54]: The adversaries do not require access to the internal parameters or structure of the target model but instead, they only rely on its output for pre-collected data samples.

Problem: Model extraction attacks severely threaten the intellectual property of GNNs. To mitigate model extraction attacks, *model ownership verification* [2], [18], [45], [61] is a very useful method. Given a suspect model $F_?$, model ownership verification determines whether $F_?$ is extracted from the target model F or is an *independent model* (precisely, a model independently trained by a benign third party). In this paper, we explore model ownership verification for GNNs. For better practicability, we focus on methods that operate under the block-box setting where the verifier can query $F_?$ and F for final output (e.g., posterior probability) but cannot access more intermediate outcomes (e.g., embedding) or model internals. We further require the method to carry no assumptions restricting the attacker's behaviors.

Study: We are not the first to study model ownership verification for GNNs. Several existing methods [49], [57], [64] can be applied directly or adapted easily to work in black-box settings. However, through empirical studies, we found that these methods either mandate assumptions that restrict the attackers or fail to provide satisfactory accuracy.

One line of methods [57], [64], denoted as **BboxVe**, adapt backdoor attacks to watermark the target model F. Briefly, **BboxVe** plants backdoors into random nodes on the training graph of F and assigns new, elaborated labels to them. If a suspect model $F_?$ presents an accuracy higher than a threshold on the backdoor nodes, $F_?$ is considered an extracted model. We reproduce the evaluation of **BboxVe** presented in [57] (detailed in §4.1). It turns out that **BboxVe** works well under a condition: *the backdoor nodes are accessed and reused by the attacker for model extraction*. Without this condition, the performance of **BboxVe** is close to random guess (see Table 3). To sum up, the success of **BboxVe** requires the attacker's "cooperation", limiting its applicability in more practical scenarios.

Another line of methods exploits the fingerprinting (or inherent behaviors) of the target model for ownership verification. **GrOVe** [49] is the representative method. It self-builds local independent models and shadow surrogate models to respectively mimic independent models in the wild and models extracted by adversaries. The two groups of models are used to train an ownership verification classifier C_{own}, which, provided the prediction output of a data point,

reports whether the output comes from a real independent model or a real surrogate model. Given a suspect model $F_?$, its output for a set of nodes is obtained and tested with C_{own}. If over 50% of nodes are predicted to be from a surrogate model, $F_?$ is considered extracted. By design, C_{own} aims at white-box ownership verification and uses the embedding of nodes as the output. We adapt it to the black-box setting by alternatively using the posterior probability as the output and name the adapted method **BGrOVe**. To assess **BGrOVe**, we replicate the evaluation presented in [57] (detailed in §4.2). It shows that **BGrOVe** presents random guess in most cases (see Table 4), failing to offer ownership verification

Method: Motivated by our empirical studies, we aim to design new, black-box methods for GNN ownership verification without unrealistic conditions attached. We follow the fingerprinting-based principle of **BGrOVe**, considering that the watermarking-based idea adopted by **BboxVe** cannot get rid of the aforementioned condition (see §5.1).

At a high level, our method also involves local independent models and shadow surrogate models to train an ownership verification classifier. However, it incorporates two insights to boost the accuracy. ❶ Our classifier considers the overall behaviors of a model instead of its output for individual data samples to make decisions. This way, the classifier utilizes holistic fingerprinting of the model rather than its local behaviors, better distinguishing independent models and surrogate models that carry less-evident differences. ❷ We enhance the fingerprinting of the target model by masking a subset of features of all nodes in its training data. This maneuvers the influence of those features on the output of the target model, thus injecting extra fingerprinting. The extra fingerprinting helps with ownership verification because it never shows up in independent models but will propagate to the surrogate models.

Evaluation: To measure the performance of our new method, we run an intensive series of evaluations, involving 5 popular datasets, 5 mainstream GNN architectures, and 16 different settings. The results show that our method can provide very high or even perfect verification accuracy in all settings while only incurring a marginal impact on the target model accuracy (0% - 5% of accuracy drop). Compared to the existing methods, our method not only offers much higher accuracy but also significantly extends their applicable scopes. We further evaluate the robustness of our method against three types of adversarial evasions (fine-tuning, double extraction, and pruning). It shows that our method successfully maintains its accuracy under all three evasion attempts.

Contributions: Our main contributions are as follows.

- We perform empirical studies and analysis to understand the performance of existing GNN ownership verification methods, unveiling that those methods fall short under the black-box settings.
- We introduce two new insights to enhance fingerprinting-based GNN ownership verification and design a new method incorporating the two insights to offer ownership verification under black-box settings.
- We run intensive evaluations of our new method. The results show that our method offers high accuracy in all kinds of settings and presents robustness against the existing evasion attempts. The source codes are released at https://github.com/rkzhou/GNN_OwnVer.

2. Background

2.1. Graph Neural Networks

Graph Neural Networks (GNNs) work on graphs. A graph $\boldsymbol{G}$ can be described as (V, E, A), where V means the node set with $v_i \in V$ representing a single node, E stands for the edge set with $e_{ij} = (v_i, v_j)$ representing a single edge, and $A \in \mathcal{R}^{|V|*|V|}$ denotes the adjacency matrix with $a_{ij} \in A$ setting to 1 if $e_{ij} \in E$ or 0 otherwise. A node v_i can also carry a feature vector $x_i \in X$ and a ground truth label $y_i \in Y$.

GNNs take G's adjacency matrix A and node features X as input and generate the embedding (or final representation) $h_{v_i} \in H$ for each node v_i. The embedding can be further converted into posterior probability via linear transformation (e.g., multilayer perceptrons) and softmax for downstream tasks like node classification [11], [15], [22], [26], [48], [58] or graph classification [7], [23], [62]. Following the settings of previous research on ownership verification for GNNs [49], **we focus on node classification** where the GNNs predict labels of given nodes.

Overview: Modern GNNs mostly adopt the method of *neighborhood aggregation* where the representation of a node is progressively updated. Specifically, in each iteration, the representation h_{v_i} of node v_i is updated by combining h_{v_i}'s current representation and the aggregated representations of its neighbors. Formally, the procedure is as follows:

$$h^l_{v_i} = Update(h^{l-1}_{v_i}, Aggregate(\{h^{l-1}_{v_j} | v_j \in \mathcal{N}(v_i)\})) \quad (1)$$

where $h^l_{v_i}$ stands for the representation of node v_i after l iterations with $h^0_{v_i}$ initialized using v_i's feature vector, and $Update(\cdot)$ and $Aggregate(\cdot)$ are parameterized functions with parameters learned through training.

Architectures: Depending on the design of $Update(\cdot)$ and $Aggregate(\cdot)$, GNNs have five popular architectures:

▶ *Graph Convolutional Network (GCN).* GCN [22] is inspired by convolutional neural networks. It combines $Update(\cdot)$ and $Aggregate(\cdot)$ into the same function, formalized as below:

$$H^l = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{l-1}W^{l-1}) \quad (2)$$

Specifically, $\tilde{A}$ (= $A + I$) is the adjacency matrix with self-connections. $\tilde{D}$ is the degree matrix of $\tilde{A}$ and $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ normalizes $\tilde{A}$. W, representing the combined function, is a layer-specific, trainable weight matrix to aggregate the information from neighbors. $\sigma(\cdot)$ is a non-linear activation function such as ReLU.

► *Simple Graph Convolution (SGC)*. Hypothesizing that the majority of the benefit of GCN arises from the local averaging and the nonlinearity between GCN layers is non-essential, SGC [55] removes the nonlinear transition function between two layers to produce a linear model. Simply speaking, SGC can be described as:

$$H^l = (\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}})^K H^{l-1}W^{l-1} \tag{3}$$

where K is the number of hops.

► *Graph Attention Networks (GAT)*. GAT [48] has an architecture similar to GCN, except that GAT leverages attention mechanisms [47] to adjust the weights of neighbors based on their similarity to the current node v_i. For example, GAT with only one attention head can be formalized as:

$$h^l_{v_i} = \sigma(\alpha_{ii}Wh^{l-1}_{v_i} + \sum_{v_j \in \mathcal{N}(v_i)} \alpha_{ij}Wh^{l-1}_{v_j}) \tag{4}$$

where a_{ij} is the attention coefficient derived from the similarity between v_i and v_j, W is the linear transformation weight matrix, and $\sigma(\cdot)$ is a non-linear activation function.

► *GraphSAGE*. GraphSAGE [15] first uses an aggregator to combine the neighbors' information and then concatenates the combined results with the current node:

$$h^l_{v_i} = Concat(h^{l-1}_{v_i}, Aggregate(\{h^{l-1}_{v_j} | v_j \in \mathcal{N}(v_i)\})) \tag{5}$$

The $Aggregate(\cdot)$ function can be a mean aggregator [15], LSTM aggregator [16], or pooling aggregator [15].

► *Graph Isomorphism Network (GIN)*. Based on Xu et. al. [58], GraphSAGE is a special case of the Weisfeiler-Lehman (WL) test [25]. Accordingly, they generalize the idea and propose GIN—an architecture as powerful as WL test for graph isomorphism:

$$h^l_{v_i} = MLP^l((1+\varepsilon^l)\cdot h^{l-1}_{v_i} + \sum_{v_j \in \mathcal{N}(v_i)} h^{l-1}_{v_j}) \tag{6}$$

where ε^l can be a trainable parameter or a fixed scalar to adjust the weight of $h^{l-1}_{v_i}$, and $MLP(\cdot)$ represent multi-layer perceptrons (MLPs) to aggregate the information of $h^{l-1}_{v_i}$ and v_i's neighbors.

Training Paradigms: In practice, GNNs can follow two different training paradigms:

►*Transductive Training*. In this setting, a GNN model is trained and tested on the same graph $G(V, E, A)$ [22], [37], [62]. The nodes for testing, which are not labeled, are still involved in the aggregation operation but not used in the loss computation.

►*Inductive Training*. In this setting, a GNN model is trained on one graph $G(V, E, A)$ and tested on another unseen graph $G'(V', E', A')$ [15], [48], [58].

Given a GNN model trained under both paradigms, the testing of a node v_i will require all nodes' features and the entire G (or G' under inductive settings) as input.

2.2. Model Extraction Attacks

Given a *target model* F, model extraction attacks [31], [32], [43] aim to reconstruct a *surrogate model* F_s to approximate F based on prior knowledge about F and queries to F. In the context of GNNs where F is trained on graph G = (V, E, A) plus node features X, the existing attacks [6], [41], [54] assume that the adversary has a surrogate graph G_s = (V_s, E_s, A_s) with node features X_s but no labels. When training F_s, the adversary queries F with data from G_s to obtain output O for target tasks like node classification and minimizes the loss between O and F_s's output O_s:

$$minimize\ \mathcal{L}(O, O_s) \tag{7}$$

where $\mathcal{L}(\cdot)$ is a loss function such as cross-entropy loss or root mean square error loss. Contingent on the responses of F to the queries, output O can be node embedding matrix H, predicted posterior probability matrix Θ, t-SNE projection matrix Υ, or simply the labels. Depending on the adversary's knowledge about F, the architecture and number of hidden layers of F and F_s may or may not be the same.

Attack Paradigms: Given different relations between G and G_s, model extraction attacks can have two paradigms:

► *Inductive Model Extraction*. In this paradigm, G_s and G are disjoint, independent graphs, but they are assumed to follow similar distributions [41].

►*Transductive Model Extraction*. In this paradigm, G_s is a sub-graph of G (or G itself), and the nodes on G_s are assumed to be involved in the training of F [6], [54].

2.3. Model Ownership Verification

Model ownership verification [3], [19], [21], [24], [29], [36], [51], [63], [65] is a mitigation against model extraction attacks. In a practical setting as introduced by Waheed et. al. [49], the verification process involves three actors: $Accuser$ (the owner of the target model F), $Responder$ (the owner of a suspect model $F_?$), and $Verifier$ (a trusted verification agent). Before F is developed, $Accuser$ is required to register it to $Verifier$ with a secure timestamp. When $F_?$ comes to attention, $Accuser$ requests $Verifier$ to verify whether $F_?$ was derived from F via model extraction attacks. In the context of GNN, attempts have been made toward model ownership verification under both black-box [57], [64] and white-box [49] settings.

Black-box Verification: In this setting, F and $F_?$ are opaque to $Verifier$. Instead, $Verifier$ can query the two models with data samples and obtain the prediction outcomes (e.g., labels or posterior probability). To support ownership verification under this setting, the existing methods [57], [64], which we name **BboxVe**, adapt backdoor attacks to watermark F. Given node classification as the downstream task, the idea is to randomly pick nodes on the training graph and watermark them by manipulating their features (e.g., changing some features to a fixed value) and assigning new, desired labels to them. F will then be trained on the

TABLE 1. IMPORTANT NOTATIONS USED IN THIS PAPER.

Notations	Descriptions
$\boldsymbol{G}$ $\boldsymbol{E}$,$\boldsymbol{V}$,$\boldsymbol{A}$ $\boldsymbol{X}$	G refers to a graph where E, V, and A respectively represent the graph's edges, nodes, and adjacency matrix. X means features of the nodes (see §2.1)
$\boldsymbol{F}$	The target model (see §2.2)
$\boldsymbol{F_s}$	The surrogate model extracted from F by attacker (see §2.2)
$\boldsymbol{F_?}$	The suspect model. In real attacks, $F_?$ = F_s (see §2.3)
$\boldsymbol{F_{wm}}$	The target model trained with watermarked nodes (see §2.3)
$\boldsymbol{F_s^*}$	Shadow surrogate models extracted by owner of F (see §2.3)
$\boldsymbol{F_{ind}}$	Independent models trained by owner of F (see §2.3)
$\boldsymbol{C_{own}}$	A classifier to detect surrogate models (see §2.3)
$\boldsymbol{F_{ind}^{train}}$	Independent models to train C_{own} (see §4.2)
$\boldsymbol{F_{ind}^{test}}$	Independent models to test C_{own} (see §4.2)
$\boldsymbol{F_{mask}}$	The target model trained with masked data (see §5.2)

watermarked graph to derive F_{wm} for public service. During verification, $F_?$ is queried with the watermarked nodes, and its accuracy on those nodes is measured using the watermarked labels as the ground truth. If the accuracy exceeds a threshold, $F_?$ is considered an extracted model, since an *independent model* (i.e., a model trained independently by a benign third party) shall present random behaviors and, thus, low accuracy on those nodes.

White-box Verification: In this setting, F and $F_?$ are assumed to be shared with $Verifier$. Waheed et. al. [49] proposed **GrOVe**, the first method of ownership verification for GNNs under this setting, considering node classification as the downstream task. They exploit model fingerprinting, namely using intrinsic model features to distinguish surrogate models from independent models. Technically, the $Accuser$ prepares a set of independent models (notated as F_{ind}) and a set of shadow surrogate models (notated as F_s^*) through local training. F_{ind} and F_s^* are then used to train an ownership classifier C_{own}, where the features are the distances between the **embedding** generated by that model for a designated dataset and the **embedding** produced by F for the same dataset. Given C_{own}, $Verifier$ can test if $F_?$ is surrogate or independent.

2.4. Important Annotations

For easy reference, important annotations used in this paper are summarized in Table 1.

3. Threat Model

In this paper, we revisit model ownership verification for GNNs. We assume that the aforementioned verification scheme, which involves $Accuser$, $Verifier$, and $Responder$, is adopted. Concerned that white-box verification is less realistic due to its requirement of $Accuser$ and $Responder$'s cooperation to share their models, we focus on black-box verification where $Accuser$ and $Responder$ only

TABLE 2. DATESETS USED IN OUR STUDY

Dataset	# Nodes	# Edges	# Features	# Classes
Cora [60]	2,708	10,556	1,433	7
CiteSeer [10]	4,230	10,674	602	6
Amazon [30]	7,650	238,162	745	8
DBLP [33]	17,716	105,734	1,639	4
PubMed [39]	19,717	88,648	500	3

deploy F and $F_?$ for remote queries. Precisely, we assume that F and $F_?$ return the predicted *posterior probability* to a query, following the common setup in the broad ownership verification literature [4], [14].

We assume that the $Accuser$ is benign, meaning it does not aim to maliciously claim that $F_?$ is extracted from F. In fact, as clarified by Waheed et. al. [49], the requirement of registering F by $Accuser$ to $Verifier$ with a timestamp will enforce this property. For generality, we do not make assumptions about the model architectures (GCN, GAT, GraphSAGE, GIN, etc.) or the training paradigm (transductive or inductive) that $Accuser$ will adopt for F.

We further assume that the $Responder$ might obtain $F_?$ through extraction attacks on F. To align with the verification scheme, the extraction attacks are assumed to be performed using data samples decided by $Responder$ and the posterior probability output by F for those samples. We do not restrict the knowledge $Responder$ has about F: It may or may not know the architecture and the number of hidden layers. In addition, $Responder$ may run transductive or inductive model extraction.

4. Motivating Studies

As overviewed in §2.3, we are not the first one to explore black-box ownership verification for GNNs. Some existing methods [57], [64] are applicable directly. Further, certain white-box methods (e.g., [49]) can be easily adapted for use. However, we found that these methods either *mandate conditions that limit their practicality* or *fail to provide satisfactory accuracy*. In this section, we unveil our findings through empirical studies and thus, motivate the need for our research: *accurate black-box GNN ownership verification methods with no strings attached*. Depending on the designs, the existing methods are either watermarking-based or fingerprinting-based, which we discuss separately.

4.1. Watermarking-based Methods

BboxVe [57], detailed in §2.3, is the only watermarking-based black-box method. A key condition required by **BboxVe** is **Condition 1:** ***the graph with backdoors injected is reused by the attacker to support model extraction***. This condition is very strong as it mandates the owner to publicize its graph with backdoors and the attacker's "cooperation" to use that graph. Removing the condition makes the method more practical, which, however, fails the verification as we will show below.

TABLE 3. EVALUATION RESULTS OF WATERMARKING-BASED OWNERSHIP VERIFICATION (**BBOXVE**). "WITHOUT BACKDOOR" MEANS THE SURROGATE MODEL IS EXTRACTED USING THE ORIGINAL GRAPH WITHOUT BACKDOOR (SETTING ❶ IN §4.1); IN CONTRAST, "WITH BACKDOOR" INDICATES THE SURROGATE MODEL IS EXTRACTED USING THE GRAPH WITH BACKDOOR (SETTING ❷ IN §4.1). **TCA**, **ECA**, **TBA**, AND **EBA** RESPECTIVELY REFER TO (1) CLEAN ACCURACY OF THE TARGET MODEL (I.E., F), (2) CLEAN ACCURACY OF THE SURROGATE MODEL (I.E., F_s), (3) BACKDOOR ACCURACY OF THE TARGET MODEL, AND (4) BACKDOOR ACCURACY OF THE SURROGATE MODEL. BACKDOOR ACCURACY ABOVE THE VERIFICATION THRESHOLD, INDICATING EXTRACTION ATTACKS, IS MARKED WITH RED OR RED OTHERWISE.

Dataset	GCN								GAT								GraphSAGE							
	With Backdoor (%)				Without Backdoor (%)				With Backdoor (%)				Without Backdoor (%)				With Backdoor (%)				Without Backdoor (%)			
	TCA	ECA	TBA	EBA	TCA	ECA	TBA	EBA	TCA	ECA	TBA	EBA	TCA	ECA	TBA	EBA	TCA	ECA	TBA	EBA	TCA	ECA	TBA	EBA
Cora	86.46	88.12	11.94	95.46	86.46	88.62	11.94	7.57	85.63	85.30	10.28	94.98	85.63	85.42	10.28	9.95	85.65	84.27	10.18	96.41	85.65	84.06	10.18	3.78
CiteSeer	78.31	76.70	16.22	98.25	78.31	75.96	16.22	7.00	79.97	79.94	14.94	97.57	79.97	79.77	14.94	7.42	74.82	73.50	13.97	98.75	74.82	72.15	13.97	5.72

Experimental Setup: In this study, we reproduce and evaluate **BboxVe**, following the method and setup described in [57]. Specifically, we focus on node classification and the **transductive training paradigm**. We reuse the datasets covered by that paper (Cora and CiteSeer as detailed in Table 2) and we consider all architectures studied in that paper (GCN, GAT, and GraphSAGE). The nodes of each dataset are split into three parts: D_{train} (20%) for training the target model F, D_{surr} (40%) for extracting the surrogate model F_s, and D_{test} (40%) for testing F and F_s.

To insert the backdoors, we randomly pick 15% of nodes from D_{train} and randomly manipulate 35 of their features to a pre-determined fixed value, as suggested by [57]. We further assign each node with the label of the least-represented class in the entire dataset. Via a similar process, we plant backdoors into 10% of nodes from D_{test} for testing. For clarity, we use F_{wm} to represent the target model trained with backdoors, which is released for public access.

We follow §2.2 to perform model extractions. Similar to [57], we configure F_s to share the same architecture and parameters (i.e., hidden layer number and hidden layer dimensions) with F_{wm}. The extraction follows Equation 7, using predicted posterior probability matrix Θ from F_{wm}/F_s as O/O_s. We repeat the attack twice under two settings: ❶ the attacker uses the original graph without backdoors for model extraction[1] and ❷ the attacker uses the original graph with backdoors injected into D_{train} for model extraction. Under the two settings, the outputs for D_{surr} are different, leading to different-behaving extracted models.

To verify the ownership of F_s, we measure two metrics of F_s on D_{test}, including (1) **backdoor accuracy**: accuracy of F_s on backdoor samples in D_{test}, using the backdoor labels as ground truth and (2) **clean accuracy**: accuracy of F_s on non-backdoor samples in D_{test}. If the clean accuracy of F_s approximates F_{wm} while the backdoor accuracy of F_s is above a threshold[2], we consider F_s an extracted model.

Evaluation Results: Following the settings above, we repeat the experiments five times and summarize the average results in Table 3. Under both settings ❶ and ❷, the clean accuracy of F_s approximates or even exceeds F. The cases where F_s outperforms F are also reported in [57], possibly because the extraction reduces model overfitting. In contrast, the backdoor accuracy of F_s is contingent on the settings. In setting ❶ where the backdoors are involved in model extraction, F_s presents a backdoor accuracy way above the threshold (similar results in all five experiments), leading to 100% detection of model extractions. Further, Welch's t-test [53] on the five experiments shows that F and F_s have statically significant differences in backdoor accuracy, indicating a high verification confidence. However, in setting ❷ where backdoors are not used during model extraction, F_s's backdoor accuracy stays substantially below the threshold, and Welch's t-test shows no differences between F and F_s. These evidence that the verification failed.

To sum up, Condition 1 **is critical to BboxVe's success**. Fundamentally, this is because watermarking only affects the target model behavior on the backdoor nodes. When those nodes are not involved in model extraction, the watermarks are not propagated to the surrogate model and the verification fails correspondingly. Hence, in principle, adopting the watermarking-based strategy cannot get rid of Condition 1.

4.2. Fingerprinting-based Methods

No existing fingerprinting-based methods operate under black-box settings. However, **GrOVe**, a white-box method described in §2.3, can be adapted to work in a black-box manner. Instead of using the embedding obtained from involved models (F, $F_?$, F_{ind}, and F_s^*) to derive features for training and testing the ownership verification classifier C_{own}, we use the predicted posterior probability output by those models. We refer to the adapted method as **BGrOVe**.

The evaluation setting of [49] and the public implementations [1] include three conditions, under which **GrOVe** presents extraordinary accuracy:

Condition A: *the **data samples** used by Accuser to extract the shadow surrogate models F_s^* are also used by the attacker to extract the real surrogate model F_s.*
Condition B: *the **number of hidden layers** in F_s^* is the same as the number of hidden layers in F_s.*
Condition C: *the **architectures** used by one or more models in F_s^* cover the architecture adopted by F_s.*

Accordingly, we first study the performance **BGrOVe** given the three conditions. To further understand **BGrOVe** in more practical settings, we remove the conditions one by one and re-perform the study.

Experimental Setup: Our study follows the setup of [49]. We focus on **node classification** and the **inductive training paradigm**, using all the five datasets summarized in Table 2

1. Recall §2.1 that testing any node requires the graph as a part of input.
2. Based on [57], the thresholds for Cora/CiteSeer are 50%/53% with GCN, 51%/48% with GAT, and 48%/49.5% with GraphSAGE.

TABLE 4. EVALUATION RESULTS OF FINGERPRINTING-BASED OWNERSHIP VERIFICATION (**BGROVE**) WITH AND WITHOUT CONDITION A. SETTINGS I, II, III, AND IV ARE EXPLAINED IN §4.2. THE TABLE SHOWS 3 METRICS OF OWNERSHIP VERIFICATION: FPR (FALSE POSITIVE RATE), FNR (FALSE NEGATIVE RATE), AND ACC (ACCURACY). FPR AND FNR ABOVE 10% ARE MARKED AS RED. THE NUMBERS FOLLOW THE FORMAT OF Ave_{std} WHERE Ave MEANS AVERAGED VALUE AND std MEANS STANDARD DEVIATION ACROSS ALL CONFIGURATIONS AND REPEATED TESTS.

With Condition A Satisfied (both F_s and F_s^* are extracted with D_{surr})

Dataset	Setting I (%)			Setting II (%)			Setting III (%)			Setting IV (%)		
	FPR	FNR	ACC	FPR	FNR	ACC	FPR	FNR	ACC	FPR	FNR	ACC
Cora	0.00±0.0	0.00±0.0	100.00±0.0	0.00±0.0	0.00±0.0	100.00±0.0	0.00±0.0	33.33±57.7	83.33±28.9	0.00±0.0	16.67±28.9	91.67±14.4
CiteSeer	0.00±0.0	77.78±38.5	61.11±19.2	0.00±0.0	88.89±19.2	55.56±9.6	0.00±0.0	66.67±57.7	66.67±28.9	0.00±0.0	100.00±0.0	50.00±0.0
Amazon	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0	20.00±0.0	83.33±28.9	48.33±14.4
DBLP	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0
PubMed	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0

With Condition A Not Satisfied (F_s are extracted with D_{surr} and F_s^* are extracted with D_{shadow_surr})

Dataset	Setting I (%)			Setting II (%)			Setting III (%)			Setting IV (%)		
	FPR	FNR	ACC	FPR	FNR	ACC	FPR	FNR	ACC	FPR	FNR	ACC
Cora	0.00±0.0	0.00±0.0	100.00±0.0	0.00±0.0	0.00±0.0	100.00±0.0	0.00±0.0	16.67±28.9	91.67±14.4	0.00±0.0	16.67±28.9	91.67±14.4
CiteSeer	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0
Amazon	0.00±0.0	77.78±38.5	61.11±19.2	0.00±0.0	88.89±19.2	55.56±9.6	3.33±2.9	83.33±28.9	56.67±13.8	40.00±13.2	33.33±57.7	63.33±22.4
DBLP	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0
PubMed	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0	0.00±0.0	100.00±0.0	50.00±0.0	1.67±2.9	100.00±0.0	49.17±1.4

and including all five architectures explained in §2.1. We split each dataset into four subsets by selecting the same ratio of random nodes for each subset: D_{train} (30%) for training the target model F and the independent models F_{ind}, D_{shadow_surr} (30%) for extracting the shadow surrogate models F_s^*, D_{surr} (30%) for extracting the real surrogate model F_s, and D_{test} (10%) for testing all models. We also disconnect edges among the four subsets to form four graphs for accommodating the inductive setting, reusing the strategy adopted by [49]. Two groups of F_{ind} are needed to train and test the ownership classifier, which we respectively refer to as F_{ind}^{train} and F_{ind}^{test}. Further, we follow the same process in §4.1 to extract F_s^* and F_s.

To train the ownership classifier C_{own}, we rely on D_{train}. For each data point dp in D_{train}, we turn it into a series of training samples. ❶ We calculate the distance between F and each model in F_{ind}^{train} on their predicted posterior probability for dp, deriving a set of *negative samples* with the distance as the feature. ❷ Similarly, we get a set of *positive samples* by calculating the distance between F and each model in F_s^* on dp. Negative and positive samples obtained from the entire D_{train} are then used to train C_{own}. Following [49], C_{own} is configured as a two-layer MLP whose best hyperparameters are obtained using a grid search. The hidden layer dimensions considered in the search are 64 or 128, and the activation function is Tanh or ReLU. Given $F_?$ for testing, we get the aforementioned distance between $F_?$ and F for each sample in D_{train} and run C_{own} to determine whether the sample is negative or positive. If **over half of the samples** in D_{train} are reported as positive by C_{own}, $F_?$ is considered extracted.

We perform the experiment twice, respectively **with** Condition A (i.e., F_s^* are extracted with D_{surr}) and **without** Condition A (i.e., F_s^* are extracted with D_{shadow_surr}). To train C_{own}, we configure F, F_{ind}^{train}, and F_s^* to choose from GCN, GAT, and GraphSAGE, all with 2 hidden layers and 256 dimensions in each layer. To test C_{own}, we diversify F_{ind}^{test} and F_s to create four settings, depending on whether Condition B and Condition C are held:

▶**Setting I** (Condition B ✓; Condition C ✓): This setting replicates [49]. In each test, we fix the target model F to use one architecture, GCN or GAT or GraphSAGE. We configure the shadow surrogate model F_s^* and training independent models F_{ind}^{train} to enumerate GCN, GAT, and GraphSAGE, all with 2 hidden layers and 256 dimensions for each layer. For each combination of architecture and hidden layer number, we train two different F_s^* and two different F_{ind}^{train} by randomizing their initial parameters[3]. In total, 6 F_s^* and 6 F_{ind}^{train} are used for training C_{own}. We also set up the real surrogate model F_s and the testing independent models F_{ind}^{test} to enumerate GCN, GAT, and GraphSAGE, all with 2 hidden layers and 256 dimensions for each layer. 6 F_s and 6 F_{ind}^{test} are used for testing C_{own}.
▶**Setting II** (Condition B ✗; Condition C ✓): This setting inherits the first setting except that we change the number of hidden layers in F_s and F_{ind}^{test} to 3.
▶**Setting III** (Condition B ✓; Condition C ✗): In this setting, we configure F_s and F_{ind}^{test} to use GIN or SGC. These two architectures are less used, which can better represent more challenging attack scenarios. The number of hidden layers in all models is kept at 2.
▶**Setting IV** (Condition B ✗; Condition C ✗): It combines the hidden layer configuration of **Setting II** and the architecture configuration of **Setting III** for F_s and F_{ind}^{test}.

Evaluation Results: We repeat each experiment five times and measure the verification accuracy of **BGrOVe** with three metrics: false positive rate (FPR, namely the portion of independent models wrong detected as surrogate models), false negative rate (FNR, namely the portion of surrogate models wrong detected as independent models), and accuracy ($\frac{TP+TN}{TN+FP+TP+FN}$). The results averaged over all config-

3. The evaluation in [49] only trains one shadow surrogate model for each combination. The reason for using a small number of models is the ownership verification classifier takes data samples as input. Thus, even a single model, when applied to D_{train}, produces sufficient training data.

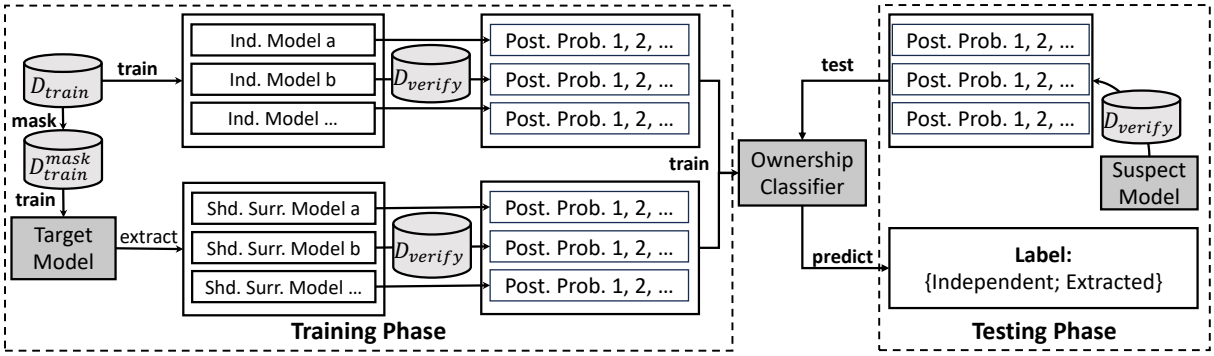


Figure 1. High-level design of our method. "Ind. Model", "Shd. Sur. Model", and "Post. Prob." are short for Independent Model, Shadow Surrogate Model, and Posterior Probability, respectively.

urations and repeated tests are summarized in Table 4[4]. For clarity, all the target models present acceptable accuracy as shown in Table 5.

On all datasets except for Cora and in most settings, **BGrOVe** presents an accuracy close to random guess, showing that **BGrOVe** prevalently fails the ownership verification task. Zooming into the results in Table 4, the low accuracy is mainly attributed to the extremely high FNR (100% in many cases). This indicates that **BGrOVe** mistakenly considers surrogate models as independent models, which offers a key insight to derive our method (see §5.1). To sum up, ***BGrOVe*** *is ill-suited for GNN ownership verification in the black-box setting*, even Conditions A, B, and C are all met.

5. Ownership Verification with Holistic, Enriched Fingerprinting

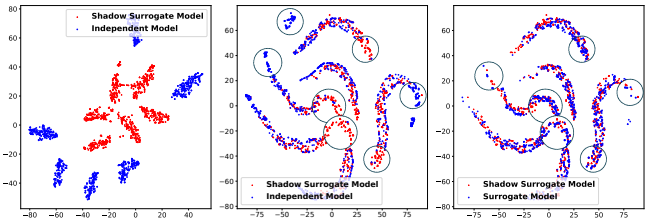
In this paper, we aim to design new, black-box methods for GNN ownership verification without additional conditions attached. As unveiled in §4, watermarking-based methods cannot get rid of Condition 1. This inspires us to focus on exploring fingerprinting-based methods.

5.1. Key Insights

At the high level, we follow the design of **BGrOVe** to create shadow surrogate models and independent models to train an ownership classifier. To escalate the accuracy, we exploit two insights.

Insight 1: The classifier used by **BGrOVe** considers model output for individual data samples to perform ownership verification (recall §4.2). This works well in white-box settings where the model output is the embedding but does not in black-box settings where the model output is the posterior probability. Using t-SNE [46], Figure 2-(a) and Figure 2-(b) visualize the output of an independent model and a shadow surrogate model for verification on the CiteSeer dataset under Condition A and Setting I. The embedding output from the two models is indeed separable (Figure 2-(a)). However, switching to the posterior probability, the two

4. For comparison, the results of using embedding as the output (i.e., the original **GrOVe**) are presented in Table 12 in Appendix.



(a) Embedding (b) Posterior Probability (c) Posterior Probability

Figure 2. The t-SNE visualization of different output for D_{train} from the CiteSeer dataset. Part (a) compares an independent model and a shadow surrogate model on embedding; Part (b) compares an independent model and a shadow surrogate model on posterior probability; Part (c) compares a shadow surrogate model and the real surrogate model on posterior probability. The circled regions in Parts (b) and (c) are where the shadow surrogate model and the independent model differ while the shadow surrogate model and the real surrogate model remain similar.

models are tangled together (Figure 2-(b)). Consequently, given the posterior probability of a sample, the classifier C_{own} faces problems predicting which model generates the output, falling short of supporting ownership verification.

While the behaviors of independent models and surrogate models on individual data samples are less distinguishable under black-box settings, we found that their *overall behaviors* still differ. As illustrated in Figure 2-(b), the shadow surrogate model and the independent model are dissimilar in both the middle and the marginal areas (the circled regions). In contrast, as shown in Figure 2-(c), the shadow surrogate model and the real surrogate model remain highly similar in those regions. This inspires us to redesign the ownership verification classifier to consider holistic model behaviors. At the high level, our classifier regards each model as a single data sample and takes its output for an entire dataset (e.g., D_{train}) as features. Given a test model, the classifier directly reports whether the model is extracted, without relying on other follow-up steps.

Insight 2: Incorporating holistic fingerprinting of the target model helps ownership verification but can remain insufficient in challenging settings where the shadow surrogate models and the real surrogate model share fewer properties (e.g., Setting IV). As we will show in §6.3, even using holistic fingerprinting, the accuracy of ownership verification can still drop below 80% under Setting IV on certain datasets.

TABLE 5. CLASSIFICATION ACCURACY OF DIFFERENT MODELS INVOLVED IN THE EVALUATION. **TARGET** REFERS TO THE TARGET MODEL (F); **INDEPENDENT** REFERS TO THE INDEPENDENT MODELS USED TO TRAIN AND TEST THE OWNERSHIP VERIFICATION CLASSIFIER (F_{ind}^{train} AND F_{ind}^{test}); **SHADOW SURROGATE** REFER TO THE SHADOW SURROGATE MODELS (F_s^*) FOR TRAINING THE OWNERSHIP VERIFICATION CLASSIFIER. THE RESULTS OF THE REAL SURROGATE MODEL ARE SIMILAR TO SHADOW SURROGATE MODELS UNDER **SETTING I**. ALL THE RESULTS ARE AVERAGED OVER DIFFERENT CONFIGURATIONS AND REPEATED EXPERIMENTS.

	Dataset	Target (%)	Independent (%)	Surrogate Model							
				Setting I (%)		Setting II (%)		Setting III (%)		Setting IV (%)	
		Accuracy	Accuracy	Accuracy	Fidelity	Accuracy	Fidelity	Accuracy	Fidelity	Accuracy	Fidelity
Inductive	Cora	67.81±1.4	63.28±4.8	63.70±3.4	72.43±2.6	63.91±3.3	72.46±2.5	63.01±5.4	70.45±3.9	62.48±5.7	69.65±4.9
	CiteSeer	80.25±0.7	78.12±1.9	79.06±0.9	87.51±1.2	79.17±0.9	87.40±1.1	78.06±1.1	86.40±1.4	77.95±1.2	86.20±1.4
	Amazon	89.18±2.6	84.03±9.0	87.65±3.0	90.01±3.4	87.86±3.0	90.15±3.4	84.93±3.2	87.54±3.7	83.31±4.9	85.59±5.2
	DBLP	72.69±1.8	69.79±2.5	72.08±1.7	83.59±2.6	72.53±1.8	83.97±2.6	70.56±2.4	81.18±2.8	70.40±2.5	80.37±3.3
	PubMed	83.61±1.2	82.35±1.9	82.86±1.6	90.91±1.9	83.10±1.5	91.04±1.9	82.43±2.3	90.23±2.4	82.50±2.2	90.19±2.4
Transductive	Cora	83.99±2.0	78.66±4.7	83.22±1.9	90.48±3.0	82.89±2.2	90.10±3.1	81.09±2.6	88.05±3.9	80.86±2.6	87.64±3.8
	CiteSeer	93.43±0.4	92.37±1.7	92.98±0.7	97.24±1.0	92.83±0.9	97.04±1.1	92.30±1.3	96.67±1.4	92.13±1.2	96.40±1.3
	Amazon	93.49±2.4	91.55±3.8	92.70±3.4	93.74±4.3	93.14±3.3	94.15±4.1	88.84±5.3	89.72±5.8	89.11±5.5	90.10±6.0
	DBLP	82.05±1.1	80.79±1.6	80.84±1.4	86.54±3.2	81.02±1.4	86.74±3.0	79.90±1.0	85.14±2.4	80.02±0.9	85.16±2.4
	PubMed	85.35±1.6	84.73±1.6	84.18±1.8	88.31±3.4	84.50±1.7	88.48±3.4	84.04±0.9	87.08±2.1	83.80±1.1	86.97±2.4

Fundamentally, the behaviors inherited by the shadow surrogate models and the real surrogate models start differing under those settings, sparking insufficient similarities to distinguish from the independent model. A visualized example demonstrating this can be found in Figure 4.

To additionally boost ownership verification accuracy, further exploring the intrinsic fingerprinting of the target model may not work. This inspires us to enrich the fingerprinting of the target model. Our idea is to "mask" the training data of the target model for injecting extra fingerprinting. Such fingerprinting is transparent to independent models but will propagate to the surrogate models, essentially facilitating ownership verification.

5.2. Method Design

Our method follows the design presented in Figure 1. It trains a binary classifier C_{own} that takes an entire model as input and predicts whether the model is extracted from the target model. The training of C_{own} has three major steps.

Enriching Model Fingerprinting: Before building the target model F, we mask its training dataset D_{train} by ***flipping a fixed subset of features of every node*** to derive D_{train}^{mask}. We then use D_{train}^{mask} to train F and release F for public service. To avoid confusion, we use F_{mask} to represent F trained with masked data. The goal of data masking is to enrich the fingerprinting of F. Precisely, we maneuver the influence of the features we pick on the posterior probability output of F, consequently injecting extra fingerprinting. Our masking is principally different from backdoor-based watermarking. First, we mask the same subset of features for all nodes, thus tuning the overall behaviors of the target model instead of its local behaviors on several nodes. Second, we do not manipulate the label of training data, better preserving the intrinsic behaviors of the target model on all data points.

A key question is what features to consider. We find that it is perfectly fine to randomly pick a subset of features to mask. What matters is *how many* features we should pick. As we will show in §6.3, masking more features will lead to a higher ownership verification accuracy but inevitably downgrade the target model performance. Fortunately, in the context of GNNs, the graph structures (e.g., connectedness) are more important than node features to model decisions. Thus, even masking a higher ratio of features can still maintain the model performance. We observe that ownership verification can become highly accurate before the masking hurts the model performance too much. For better verification accuracy, we suggest the users mask more features until the target model accuracy drops by a certain threshold (5% is sufficient based on our evaluations).

Building Local Models: To train the ownership verification classifier C_{own}, we need to train a set of local independent models and extract a set of shadow surrogate models. By intuition, the quality of the two sets of models affects the capacity and generality of C_{own}. However, we cannot make unrealistic requirements like enumerating all possible architectures and parameters that wild models or attackers may adopt. To this end, we focus on building local models that better represent challenging situations to train C_{own}. First, we extract the shadow surrogate models using samples different from the training data of the target model F_{mask}. Doing so helps C_{own} learn about surrogate models further away from F_{mask}. Second, we configure the local independent models to reuse the training settings of F_{mask} (including the number of hidden layers, the dimensions of each layer, the training method, and the training data). This creates independent models closer to F_{mask}, aiding C_{own} to identify benign models less distinguishable from F_{mask}. Finally, we diversify the two sets of models to cover popular GNN architectures (e.g., GCN, GAT, GraghSAG, etc.), improving the generality of C_{own}.

Training Verification Classifier: We run each independent model to output the posterior probability for every data sample in D_{train} (i.e., the training data of the target model). Concatenating the posterior probability together, we derive a single vector to work as the features of the independent model. The goal of merging the posterior probability is to

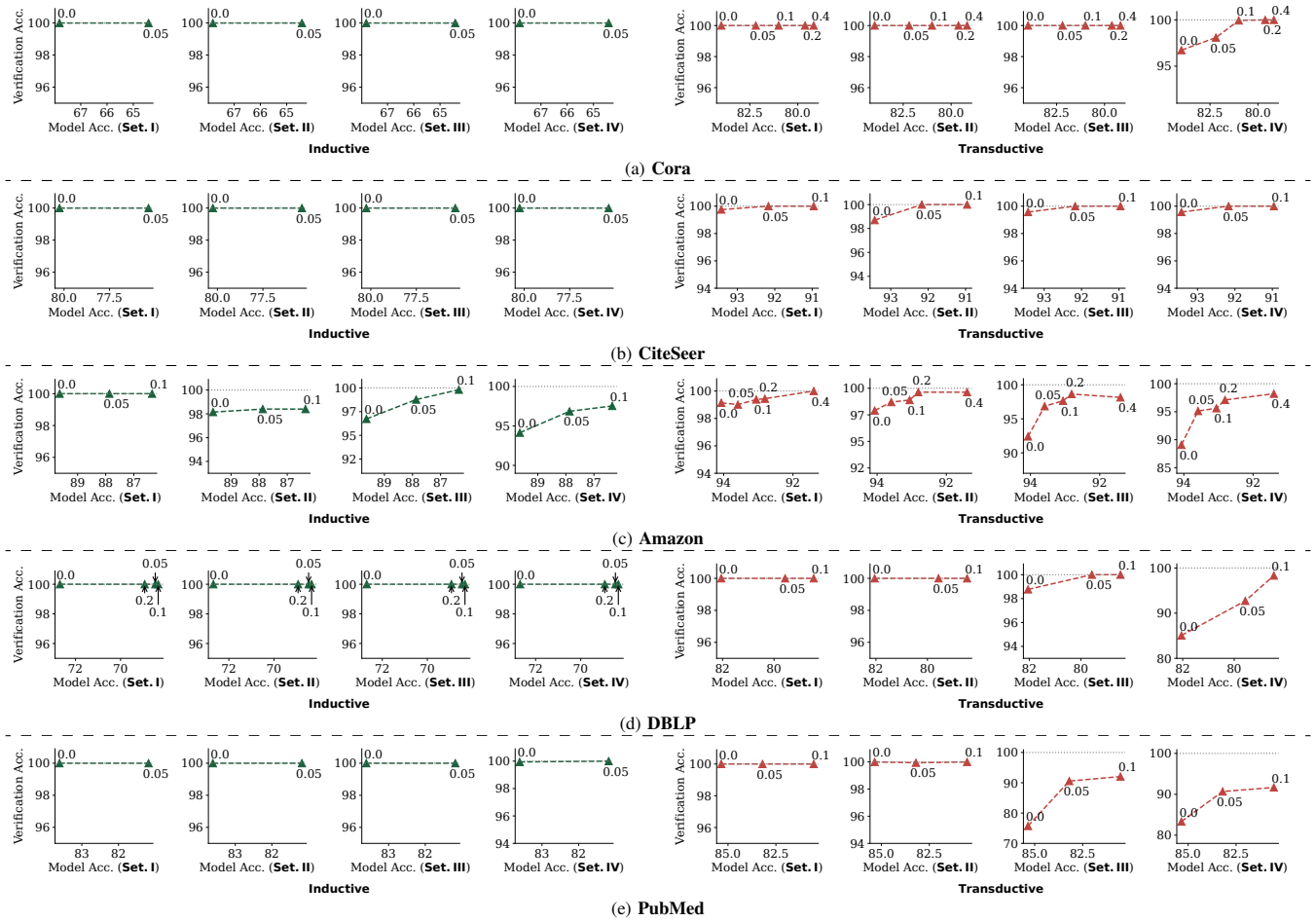


Figure 3. Change of target model accuracy and ownership verification accuracy with the increase of masking level. **The value above/below each point represents the ratio of features getting masked (e.g., 0.05 indicates 5% of all features are masked)**. The point with "0.0" indicates no masking is applied. All the results are obtained **without Condition A**, namely the shadow surrogate models and the real surrogate models are extracted using different data samples. "**Model Acc.**" refers to target model accuracy and "**Set.**" is short for **Setting**.

consider the model behaviors as a whole, thus leveraging holistic fingerprinting. Similarly, we can produce the features for every shadow surrogate model. Using the two sets of models, we train C_{own} via mini-batch training with 10 samples in each batch. The resulting C_{own} is a two-layer MLP classifier with dimensions of [128, 64] and ReLU as the activation function.

6. Evaluation

6.1. Experimental Setup

To understand the performance of our method, we perform a series of evaluations following the settings described in §4.2. We similarly repeat the evaluations under Settings I-IV with and without Condition A (recall §4.2). As our ownership verification classifier (C_{own}) takes models as data points, we increase the number of independent models (F_{ind}^{train}) and the number of shadow surrogate models (F_s^*) for training C_{own}. Specifically, we build 20 F_{ind}^{train} and 20 F_s^* for each combination of architecture and number of hidden layers. In total, we use 60 F_{ind}^{train} and 60 F_s^* to train C_{own}. We follow a similar protocol to increase the number of models to test C_{own}. In Settings I and Settings II, we use 60 F_{ind}^{test} and 60 F_s^* for testing, and in Settings III and Settings IV, we use 40 F_{ind}^{test} and 40 F_s^* for testing.

Different from §4.2, **we consider both the inductive paradigm and the transductive paradigm**. Further, our method requires masking some features of the training data of the target model. In our evaluation, we repeat the experiments by gradually increasing the ratio of random features to mask (0%, 5%, 10%, 20%, 40%, and 60%) until the target model accuracy drops by 5%.

6.2. Model Accuracy

The first evaluation question we aim to understand is *whether the GNN models involved in our evaluation present satisfactory performance* because otherwise, the verification becomes meaningless. Accordingly, we measure the accuracy ($\frac{TP+TN}{TN+FP+TP+FN}$) of different models and summarize the results in Table 5.

The target model presents expected accuracy in all cases. The results are similar to what is reported in related research [49], [57] when considering overlapped datasets and settings. The independent models for training and testing C_{Own} both share similar accuracy with the target model. The accuracy difference is mostly $\pm 2\%$. Our feature masking can indeed hurt the accuracy of the target model. In general, the accuracy drops more when more features are masked (see Figure 3 and Figure 6). However, we must highlight that our method works very well when mandating the model accuracy drop to not exceed 5%.

The results in Table 5 are averaged across Settings I-IV. In Settings III and IV, the architectures of target models (GCN, GAT and GraphSAGE) and independent models (GIN and SGC) are different. GCN, GAT and GraphSAGE have higher accuracy than GIN and SGC (see Table 11 in Appendix), making target models appear more accurate.

The surrogate models approximate the performance of the target model with a slightly reduced accuracy, which aligns with the expectations. In Setting I where the surrogate models share more properties with the target model, the accuracy decrease is very minor (less than 2% in most cases). In other settings where the properties of the model groups further diverge, the accuracy drops a bit more but still less than 5% in all cases. Further, the surrogate models present high fidelity—measured by the ratio of testing samples where the surrogate models predicted the same label as the target model. In most cases, the fidelity is over 85%.

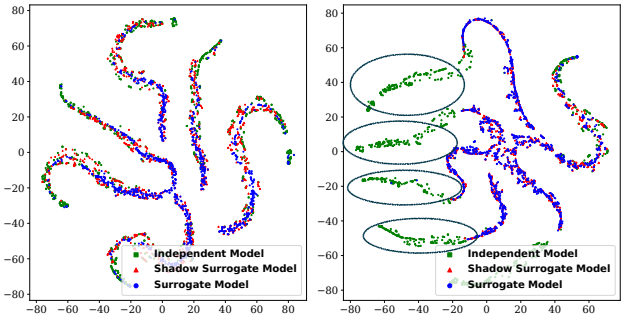
To sum up, our target models and independent models offer satisfactory accuracy. Our surrogate models present successful attacks. These provide the foundations for meaningful evaluations of ownership verification.

6.3. Ownership Verification Accuracy

The second question—also the key question—of our evaluation is to understand the accuracy and the cost of our ownership verification method. To this end, we measure the accuracy ($\frac{TP+TN}{TN+FP+TP+FN}$) of our method under different masking levels and different settings (inductive and transductive; with and without Condition A; Settings I-IV). In Figure 3 and Figure 6, we show the trade-off between target model accuracy and our ownership verification accuracy without and with Condition A respectively. FPR and FNR corresponding to the points in the two figures are summarized in Table 13.

❶ Without any masking, our method already presents extremely high accuracy in many cases. In all settings under the inductive training paradigm, our method achieves 100% accuracy on Cora, CiteSeer, DBLP, and PubMed. Under the transductive training paradigm, our method produces similar accuracy given less challenging settings (Setting I and Setting II). These results demonstrate the effectiveness of using holistic fingerprinting for GNN ownership verification.

❷ Given the transductive training paradigm and the more challenging settings (Setting III and Setting IV), our method without masking no longer provides perfect accuracy. On datasets like PubMed, our verification accuracy can drop to 75%. This is understandable because in those settings (i) the independent models use the same graph as the target model for training, leading to higher similarity between them, and (ii) the shadow surrogate models and the real surrogate models share fewer properties and are less close.



(a) No Mask (b) 20% Mask

Figure 4. The t-SNE visualization of posterior probability for D_{train} from the CiteSeer dataset in Setting IV. Part (a) compares the target model, a shadow surrogate model, and the real surrogate model **without masking**; Part (b) shows a similar comparison when **masking 20% features**. The circled regions in Part (b) highlight where the target model differs from the other two models.

Nevertheless, Our feature masking greatly helps those cases. On Cora, CiteSeer, and DBLP, masking 10-20% of the features can escalate our verification accuracy to 100%. On Amazon and PubMed, feature masking does not lead to perfect accuracy but still significantly increases the verification accuracy (up to 98% and 92%). The results showcase the success and the necessity of our feature masking in challenging scenarios. To better illustrate how feature masking helps, we visualize the posterior probability output of an independent model, a shadow surrogate model, and the real surrogate model on the CiteSeer dataset under Setting IV without Condition A, respectively without feature masking and with 20% feature masking. Evidently, our feature masking enables the shadow surrogate model and the real surrogate model to distinguish from the independent model, justifying why feature masking helps.

❸ As shown in Figure 3 and Figure 6, our feature masking is a trade-off between model accuracy and verification accuracy. Performing heavier masking leads to better verification accuracy but reduces the model accuracy. However, as we clarified in §5.2, node features play a less important role in GNN decisions and thus, masking features imposes a lower impact on model accuracy. Most critically, by only enforcing masking with limited impact on model accuracy (a $< 5\%$ drop), we can achieve perfect accuracy in nearly all cases.

A noticeable exception occurs on the PubMed dataset under the transductive training paradigm and Settings III, IV. Masking 10% of the features, we reduce the model accuracy to about 5% but only increase the verification accuracy to 92%. We need to mask around 40% of the features to reach perfect verification accuracy, incurring a 10% model accuracy decrease. Zooming into the results, we find that the verification accuracy is mainly affected by cases where

TABLE 6. ADVERSARIAL ROBUSTNESS RESULTS: IMPACT OF FINE-TUNING ON THE ACCURACY OF OUR OWNERSHIP VERIFICATION METHOD. ORI. ACC INDICATES THE VERIFICATION ACCURACY BEFORE FINE-TUNING IS PERFORMED.

	Dataset	Setting I				Setting II				Setting III				Setting IV			
		Ori. ACC (%)	Fine-Tuning (%)			Ori. ACC (%)	Fine-Tuning (%)			Ori. ACC (%)	Fine-Tuning (%)			Ori. ACC (%)	Fine-Tuning (%)		
			FPR	FNR	ACC		FPR	FNR	ACC		FPR	FNR	ACC		FPR	FNR	ACC
Inductive	Cora	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00
	CiteSeer	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00
	Amazon	100.00	0.00	0.00	100.00	98.40	3.66	0.00	98.17	99.81	0.25	0.00	99.88	97.53	5.29	0.00	97.36
	DBLP	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00
	PubMed	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.22	0.00	99.89
Transductive	Cora	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00
	CiteSeer	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00
	Amazon	100.00	0.00	0.00	100.00	99.63	0.82	0.00	99.59	98.15	3.51	2.03	97.73	98.21	0.87	4.12	97.51
	DBLP	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	98.33	1.46	0.00	99.27
	PubMed	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	92.04	16.38	0.00	91.81	91.67	16.76	0.00	91.62

TABLE 7. FALSE POSITIVE RATES OF OUR METHOD ON INDEPENDENT MODELS INTRODUCED IN OUR EXTENDED STUDY. "INDUC." AND "TRANS." ARE SHORT FOR INDUCTIVE AND TRANSDUCTIVE.

Dataset	Training	Setting I (%)	Setting II (%)	Setting III (%)	Setting IV (%)
Cora	Induc.	0.00±0.0	0.00±0.0	0.00±0.0	0.0±0.0
	Trans.	0.00±0.0	0.00±0.0	0.00±0.0	0.0±0.0
CiteSeer	Induc.	0.00±0.0	0.00±0.0	0.00±0.0	0.0±0.0
	Trans.	0.00±0.0	0.00±0.0	0.00±0.0	0.0±0.0
Amazon	Induc.	0.00±0.0	2.27±0.8	0.18±0.6	3.87±1.9
	Trans.	0.00±0.0	0.15±0.9	1.88±2.4	0.26±0.7
DBLP	Induc.	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	Trans.	0.00±0.0	0.00±0.0	0.00±0.0	2.06±4.5
PubMed	Induc.	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	Trans.	0.00±0.0	0.00±0.0	12.59±18.4	15.68±15.6

the target model uses GraphSage while the independent models and the surrogate models use other architectures. Presumably, this is because PubMed has fewer features (only 500). To reach a sufficient number of features to influence the model behaviors, a higher ratio is needed.

Extended Studies I: In the evaluations above, we only considered independent models trained using the same training dataset as the target model. While this represents a more challenging scenario where the independent models and the target model are closer, it falls short of proving our generality regarding training data. To this end, we extend the evaluations to include more independent models trained on varying data. Specifically, for each combination of architecture and hidden layer number, we further train 20 independent models with 30% of nodes randomly picked from the entire graph (the node picking is re-done for each model). In total, we create 60 extra independent models under Setting I and Setting II and 40 under Setting III and Setting IV. We also picked the masking ratio to drop the target model accuracy by about (but not over) 5%.

Table 7 shows the results of the extended study. Our method maintains its performance on those models. The false positive rate is zero in nearly all cases, showing no difference from the previous evaluations. Only in the case of PubMed under Setting III and Setting IV, our method presents a 10%+ false positive. This is also caused by the cases where the target model uses GraghSAGE. If we increase the masking to drop the target model accuracy by about 10%, all false positives disappear.

Extended Studies II: Further, the evaluation above only considered a fixed number of local models to train the ownership verification classifier. We extend it to vary the number of local models from 120 to 30, 60, 90, 120, and 150 under all different settings.

Table 14 summarizes the results of our extended study. The number of local models indeed has an impact on ownership verification: more local models increase the verification accuracy to a small extent. When we increase the local model number from 30 to 150, the increase of accuracy stays below 1% in most cases.

6.4. Adversarial Robustness

In practice, adversaries may attempt to evade the detection of our ownership verification method. To this end, the third question we aim to answer is the robustness of our method against adversarial evasions. Previous research [49] introduced four model-retraining techniques to evade ownership verification. One technique, named distribution-shift, "*could not successfully modify*" the extracted model, which we skip accordingly. In this evaluation, we measure the robustness of our method against the other three evasion techniques: *fine-tuning*, *double extraction*, and *pruning*.

Given an evasion technique, we apply it to the surrogate model to derive a new model and re-evaluate the accuracy of our ownership verification classifier on the new model. One factor affecting our verification method is the ratio of features we mask. As a validating evaluation, we focus on the masking ratio contributing to optimal verification accuracy. Specifically, we pick a masking ratio to drop the target model accuracy by about but not exceeding 5% (i.e., the point on the far right in Figure 3). Further, we focus on the most challenging case: Setting IV without Condition A.

❶ **Fine-tuning** [5], [17], [52], in the context of machine learning, refers to adjusting a pre-trained model on another dataset to improve its performance. Following [49], we perform end-to-end fine-tuning to F_s by retraining it on D_{test} (the dataset for testing the target model). The fine-tuning updates both the intermediate output (e.g., embedding) and the final output (e.g., posterior probability) of F_s.

TABLE 8. ADVERSARIAL ROBUSTNESS RESULTS: IMPACT OF DOUBLE EXTRACTION ON THE ACCURACY OF OUR VERIFICATION METHOD.

	Dataset	Setting I				Setting II				Setting III				Setting IV			
		Ori. ACC (%)	Double Extraction (%)			Ori. ACC (%)	Double Extraction (%)			Ori. ACC (%)	Double Extraction (%)			Ori. ACC (%)	Double Extraction (%)		
			FPR	FNR	ACC		FPR	FNR	ACC		FPR	FNR	ACC		FPR	FNR	ACC
Inductive	Cora	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00
	CiteSeer	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00
	Amazon	100.00	0.00	0.00	100.00	98.40	2.12	0.00	98.94	99.81	0.21	0.00	99.90	97.53	3.12	0.00	98.14
	DBLP	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00
	PubMed	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.12	0.00	99.94
Transductive	Cora	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00
	CiteSeer	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00
	Amazon	100.00	0.00	0.00	100.00	99.63	0.23	0.00	99.89	98.15	1.64	1.17	98.60	98.21	0.26	2.99	98.38
	DBLP	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	98.33	1.23	0.00	99.38
	PubMed	100.00	0.00	0.00	100.00	100.00	0.00	0.00	100.00	92.04	14.81	0.00	92.59	91.67	14.81	0.00	92.59

In Table 6, we present the evaluation results. In all settings and on all datasets, the accuracy of our ownership verification method is barely affected, indicating the robustness of our method against fine-tuning. This is not surprising. In principle, fine-tuning amends rather than eliminating the existing behaviors of F_s. Thus, the behaviors F_s inherits from the target model, including those introduced by feature masking, remain and still enable ownership verification. To effectively evade ownership verification, the fine-tuning must be intensive enough to significantly erase the inherited behaviors, which, however, would offset the value of model extraction and make the attack meaningless.

❷ **Double extraction** means the adversaries perform two rounds of model extraction attacks, with the first round against the target model to derive an intermediate model F_t and the second round against the intermediate model to produce the surrogate model F_s. The intention is to make F_s further distinct from the target model. In this evaluation, we experiment with double extraction by running the model extraction attack presented in §4.2 twice. The first attack follows §4.2 exactly to produce F_t. The second attack launches extraction against F_t using the same query dataset. It further configures F_s to use the same architecture as F_t to reduce the impact on model accuracy.

We present the evaluation results in Table 8. Similar to fine-tuning, double extraction has a marginal impact on the accuracy of our verification method. Fundamentally, for double or further duplicated extractions to preserve usable model accuracy, they must remain approximating or inheriting the behaviors of the target model, which inevitably leaves over fingerprinting for ownership verification.

❸ **Pruning** [28], [40], [67] removes certain weights from the extracted model. The intended goal of pruning is to decrease the computational complexity. However, since it alters model behaviors and output, previous research [49] considers it beneficial for evading ownership verification. Following [49], we randomly prune a certain ratio of weights from the extracted model F_s and rerun ownership verification. Specifically, we enumerate the pruning ratio from 0.1 to 0.5 until F_s accuracy drops by 10%.

The evaluation results are presented in Figure 5. In all cases except for Amazon, pruning presents negligible impacts on our verification accuracy even though it already reduces the model accuracy. This demonstrates the robustness of our method against pruning. Given Amazon, pruning 20% of the weights starts visibly decreasing the accuracy of our verification method (by about 5%) under the transductive paradigm. However, it concurrently decreases the model accuracy by 10%. This shows that to evade our ownership verification, heavy pruning must be performed, which in turn will hurt the model accuracy significantly and diminish the benefits of model extractions.

❹ **Adaptive attacks** [59] assume powerful adversaries who have the full knowledge of the verification system, including the verification classifier and the verification data. Accordingly, the adversaries optimize the surrogate models to produce outputs confusing the verification classifier when given the same verification data. We believe such adversaries can be less realistic under our threat model. First, the $Verifier$, a trusted agent, has no incentive to release the verification classifier since doing so has no extra benefits yet compromises the intellectual property of the classifier itself. Second, releasing the verification data also has no necessity in our verification scheme while offering the adversaries a universal way to bypass the verification: by sending arbitrary responses to the verification data, the adversaries will certainly interrupt any verification classifier. Nevertheless, we extend two studies as follows.

(1) We assume the adversaries have the verification classifier but not the verification data and follow the methods presented in [59] to optimize the surrogate model. Following our previous experimental setup (see §6.1), we show the false negative rates of our verification under Setting I-IV and transductive learning in Table 10. Our method is robust in this scenario.

(2) We assume the adversaries have both the verification classifier and the verification data. We repeat the study and show the results in Table 10. The attack evades our verification. This is anticipated as we explained above: The optimization "ensures" the surrogate model produces outputs that confuse the verification classifier. The method presented in [59] cannot resist the attack, either. Instead, they switched to a different, less-accurate, randomness-enabled method as mitigation.

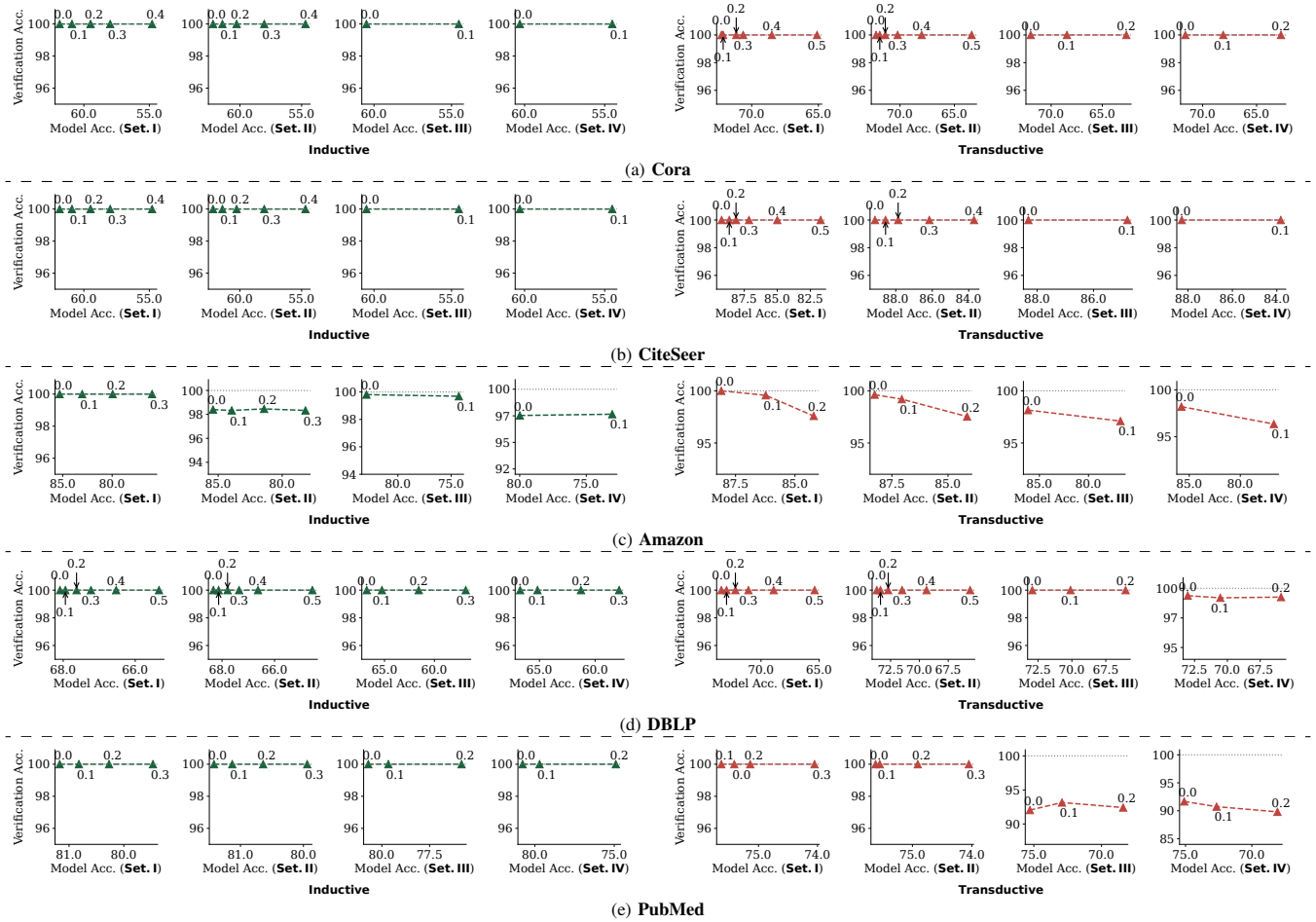


Figure 5. Adversarial robustness results: impact of model pruning on the accuracy of our ownership verification method. **The value above/below each point represents the ratio of pruning (e.g., 0.1 indicates 10% of weights are pruned)**. The point with "0.0" indicates no pruning is applied.

6.5. Efficiency

We finally assess the efficiency of our method considering that time cost is important in practice. Our method involves two phases: an *offline* phase which trains the ownership verification classifier and an *online* phase which tests a suspect model. We separately measure the time cost of the two phases. All our evaluations are performed on machines with Intel(R) Core(TM) i9-12900K (16 cores), NVIDIA GeForce RTX 4090 (24GB GPU memory), and 64GB RAM.

Table 9 in Appendix shows our evaluation results. Most of the time (over 99%) is spent in the offline phase, which only needs to run once. In the online phase, our method only requires less than 0.5 seconds to verify a suspect model, presenting high efficiency for practical deployment.

7. Discussion: Motivation and Generalization

GNNs are gaining high popularity and importance in various domains (social network analysis [8], [50], recommendation systems [9], [56], cybersecurity [35], [66], etc.). However, our studies unveil that practical while effective black-box model ownership verification remains missing for GNNs, despite the threats posed by model extraction attacks.

Our high-level idea can be applied to other neural networks, but our method exploits a unique property of graphs: the node features are less important than the graph structures for GNN decisions. Thus, we can "inject" extra fingerprinting into the nodes without hurting model accuracy much (see Section 5.2).

8. Related Work

Model Extraction Attacks. Model extraction has been extensively explored across various fields. Tramèr et. al. [43] introduce the initial work on extracting machine learning models by solely querying APIs. They proposed two attack types—equation-solving attacks and path-finding attacks—to extract logistic regressions and decision trees, respectively. Subsequent studies expanded these attacks to more intricate models like Convolutional Neural Networks [20], [32]. To reduce the dependency on original training data, Papernot et. al. [34] propose Jacobian-Based Dataset Augmentation (JBDA), which constructs a

synthetic dataset by augmenting a seed dataset iteratively. Orekondy et. al. [32] further present an attack without knowledge about training/testing data used by the target model. Simply speaking, they adopt random images from a different distribution (or a surrogate dataset) to query the target model to enable the attack. Several recent works [27], [38], [44] can even extract models without data. For example, Truong et. al. [44] develop a data-free model extraction that leverages generative adversarial networks to craft query samples and train the surrogate model to mimic the outputs of the target model.

Earlier works, as described above, primarily focus on the image domain. More recent research introduces model extractions to the context of graphs. DeFazio et. al. [6] proposed the first GNN extraction by iteratively perturbing the target model's training graph to derive multiple subgraphs for querying and extracting the target model. Shen et. al. [41] proposed the first model stealing attacks against inductive GNN by querying the target model with a surrogate dataset. More systematically, Wu et. al. [54] formalize the threat model of GNN extraction attacks and classify the threats into seven categories by considering different background knowledge of the attacker (e.g., attributes and/or neighbor connections of the nodes known by the attacker).

Model Ownership Verification. To protect the intellectual property of the model owners, researchers have explored various approaches for model ownership verification. Adi et. al. [2] and Zhang et. al. [61] adopt backdoor attacks to watermark the target model. If a suspect model presents the backdoors (usually determined by its accuracy on backdoor data), it is considered stolen from the target model. Uchida et. al. [45] propose a different strategy where they embed bit messages into the model weights as the watermark. To run ownership verification, they compare the bit message in a suspect model and the target model. To apply this method, white-box access to the suspect model is required. However, follow-up research unveils that these methods may not work well against models extracted via API queries. Fundamentally, as claimed by Jia et. al. [18], the backdoors are not sampled from the task distribution and thus, can be easily removed during model extraction. To this end, they introduce entangled watermark embedding to enable the model to concurrently learn the watermarks and how to classify samples from the task distribution.

In the context of GNNs, model ownership verification has also raised early interest. Xu et. al. [57] and Zhao et. al. [64] insert backdoors as a watermark to enable ownership verification. In contrast, Waheed et. al. [49] proposes to use the embedding output of GNN models as the fingerprint to identify the ownership of a suspect model, which, however, requires white-box access to the target model. As unveiled in §4, these methods either mandate unrealistic conditions or present unsatisfactory accuracy in the black-box setting, motivating our research presented in this paper.

9. Conclusion

In conclusion, this paper proposed a novel method that leverages local independent models and shadow surrogate models for GNN ownership verification. This method incorporates two key insights: a consideration of the overall behaviors of a suspect model for decision-making, and an enhancement of fingerprinting by masking a subset of features in the training data. Our evaluations demonstrate the effectiveness of the proposed method in improving accuracy under black-box scenarios.

10. Acknowledgment

We thank the anonymous reviewers and shepherd for their valuable feedback. This work was supported by National Science Foundation (NSF) awards CNS-2029038, CNS-2135988, and OAC-2319880. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the US government or NSF.

References

[1] "Grove: Ownership verification of graph neural networks using embeddings." https://github.com/ssg-research/GrOVe, 2023.

[2] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning your weakness into a strength: Watermarking deep neural networks by backdooring," in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1615–1631.

[3] X. Cao, J. Jia, and N. Z. Gong, "Ipguard: Protecting intellectual property of deep neural networks via fingerprinting the classification boundary," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, pp. 14–25.

[4] L. Charette, L. Chu, Y. Chen, J. Pei, L. Wang, and Y. Zhang, "Cosine model watermarking against ensemble distillation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 9, 2022, pp. 9512–9520.

[5] K. W. Church, Z. Chen, and Y. Ma, "Emerging trends: A gentle introduction to fine-tuning," *Natural Language Engineering*, vol. 27, no. 6, pp. 763–778, 2021.

[6] D. DeFazio and A. Ramesh, "Adversarial model extraction on graph neural networks," *arXiv preprint arXiv:1912.07721*, 2019.

[7] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, 2016.

[8] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The world wide web conference*, 2019, pp. 417–426.

[9] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He *et al.*, "A survey of graph neural networks for recommender systems: Challenges, methods, and directions," *ACM Transactions on Recommender Systems*, vol. 1, no. 1, pp. 1–51, 2023.

[10] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of the third ACM conference on Digital libraries*, 1998, pp. 89–98.

[11] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[12] X. Gong, Y. Chen, W. Yang, G. Mei, and Q. Wang, "Inversenet: Augmenting model extraction attacks with training data inversion." in *IJCAI*, 2021, pp. 2439–2447.

[13] X. Gong, Q. Wang, Y. Chen, W. Yang, and X. Jiang, "Model extraction attacks and defenses on cloud-based machine learning models," *IEEE Communications Magazine*, vol. 58, no. 12, pp. 83–89, 2020.

[14] J. Guan, J. Liang, and R. He, "Are you stealing my model? sample correlation for fingerprinting deep neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36 571–36 584, 2022.

[15] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018.

[18] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, "Entangled watermarks as a defense against model extraction," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1937–1954.

[19] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "Prada: protecting against dnn model stealing attacks," in *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 512–527.

[20] S. Kariyappa, A. Prakash, and M. K. Qureshi, "Maze: Data-free model stealing attack using zeroth-order gradient estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 814–13 823.

[21] M. Kesarwani, B. Mukhoty, V. Arya, and S. Mehta, "Model extraction warning in mlaas paradigm," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 371–380.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[23] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *International conference on machine learning*. PMLR, 2019, pp. 3734–3743.

[24] T. Lee, B. Edwards, I. Molloy, and D. Su, "Defending against neural network model stealing attacks using deceptive perturbations," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 43–49.

[25] A. Leman and B. Weisfeiler, "A reduction of a graph to a canonical form and an algebra arising during this reduction," *Nauchno-Technicheskaya Informatsiya*, vol. 2, no. 9, pp. 12–16, 1968.

[26] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[27] Z. Lin, K. Xu, C. Fang, H. Zheng, A. Ahmed Jaheezuddin, and J. Shi, "Quda: Query-limited data-free model extraction," in *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, 2023, pp. 913–924.

[28] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018.

[29] N. Lukas, Y. Zhang, and F. Kerschbaum, "Deep neural network fingerprinting by conferrable adversarial examples," *arXiv preprint arXiv:1912.00888*, 2019.

[30] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 43–52.

[31] D. Oliynyk, R. Mayer, and A. Rauber, "I know what you trained last summer: A survey on stealing machine learning models and defences," *ACM Computing Surveys*, 2023.

[32] T. Orekondy, B. Schiele, and M. Fritz, "Knockoff nets: Stealing functionality of black-box models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4954–4963.

[33] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, "Tri-party deep network representation," *Network*, vol. 11, no. 9, p. 12, 2016.

[34] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506–519.

[35] X. Pei, L. Yu, and S. Tian, "Amalnet: A deep learning framework based on graph convolutional networks for malware detection," *Computers & Security*, vol. 93, p. 101792, 2020.

[36] Z. Peng, S. Li, G. Chen, C. Zhang, H. Zhu, and M. Xue, "Fingerprinting deep neural networks globally via universal adversarial perturbations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 430–13 439.

[37] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[38] S. Sanyal, S. Addepalli, and R. V. Babu, "Towards data-free model stealing in a hard label setting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 15 284–15 293.

[39] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[40] S. Sen, N. Moha, B. Baudry, and J.-M. Jézéquel, "Meta-model pruning," in *International Conference on Model Driven Engineering Languages and Systems*. Springer, 2009, pp. 32–46.

[41] Y. Shen, X. He, Y. Han, and Y. Zhang, "Model stealing attacks against inductive graph neural networks," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1175–1192.

[42] T. Takemura, N. Yanai, and T. Fujiwara, "Model extraction attacks on recurrent neural networks," *Journal of Information Processing*, vol. 28, pp. 1010–1024, 2020.

[43] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction {APIs}," in *25th USENIX security symposium (USENIX Security 16)*, 2016, pp. 601–618.

[44] J.-B. Truong, P. Maini, R. J. Walls, and N. Papernot, "Data-free model extraction," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 4771–4780.

[45] Y. Uchida, Y. Nagai, S. Sakazawa, and S. Satoh, "Embedding watermarks into deep neural networks," in *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, 2017, pp. 269–277.

[46] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[48] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[49] A. Waheed, V. Duddu, and N. Asokan, "Grove: Ownership verification of graph neural networks using embeddings," in *2024 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2024, pp. 50–50.

[50] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed graph clustering: A deep attentional embedding approach," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. AAAI Press, 2019, p. 3670–3676.

[51] S. Wang and C.-H. Chang, "Fingerprinting deep neural networks-a deepfool approach," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2021, pp. 1–5.

[52] Y.-X. Wang, D. Ramanan, and M. Hebert, "Growing a brain: Fine-tuning by increasing model capacity," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2471–2480.

[53] B. L. Welch, "The generalization of 'student's' problem when several different population varlances are involved," *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947.

[54] B. Wu, X. Yang, S. Pan, and X. Yuan, "Model extraction attacks on graph neural networks: Taxonomy and realisation," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 337–350.

[55] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*. PMLR, 2019, pp. 6861–6871.

[56] S. Wu, F. Sun, W. Zhang, X. Xie, and B. Cui, "Graph neural networks in recommender systems: a survey," *ACM Computing Surveys*, vol. 55, no. 5, pp. 1–37, 2022.

[57] J. Xu, S. Koffas, O. Ersoy, and S. Picek, "Watermarking graph neural networks based on backdoor attacks," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2023, pp. 1179–1197.

[58] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[59] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li, "Detecting ai trojans using meta neural analysis," in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 103–120.

[60] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International conference on machine learning*. PMLR, 2016, pp. 40–48.

[61] J. Zhang, Z. Gu, J. Jang, H. Wu, M. P. Stoecklin, H. Huang, and I. Molloy, "Protecting intellectual property of deep neural networks with watermarking," in *Proceedings of the 2018 on Asia conference on computer and communications security*, 2018, pp. 159–172.

[62] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[63] J. Zhao, Q. Hu, G. Liu, X. Ma, F. Chen, and M. M. Hassan, "Afa: Adversarial fingerprinting authentication for deep neural networks," *Computer Communications*, vol. 150, pp. 488–497, 2020.

[64] X. Zhao, H. Wu, and X. Zhang, "Watermarking graph neural networks by random graphs," in *2021 9th International Symposium on Digital Forensics and Security (ISDFS)*. IEEE, 2021, pp. 1–6.

[65] Y. Zheng, S. Wang, and C.-H. Chang, "A dnn fingerprint for non-repudiable model ownership identification and piracy detection," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2977–2989, 2022.

[66] Y. Zhou, S. Liu, J. Siow, X. Du, and Y. Liu, "Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks," *Advances in neural information processing systems*, vol. 32, 2019.

[67] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.

Appendix A. Experimental Results

TABLE 9. TIME COST OF OUR OWNERSHIP VERIFICATION METHOD. **OFFLINE** INDICATES THE AVERAGE TIME NEEDED TO TRAIN THE CLASSIFIER, WHICH IS DONE ONCE FOR ALL. **ONLINE** SHOWS THE AVERAGE TIME COST OF VERIFYING ONE MODEL.

Dataset	Offline	Online	Online Ratio
Cora	18.13s	0.12s	0.67%
CiteSeer	14.99s	0.13s	0.87%
Amazon	29.44s	0.19s	0.65%
DBLP	52.53s	0.39s	0.74%
PubMed	35.25s	0.28s	0.79%

TABLE 10. FALSE NEGATIVE RATES OF OUR METHODS AGAINST ADAPTIVE ATTACKS IN TRANSDUCTIVE LEARNING

Attackers only have the verification classifier					
Dataset	Cora	CiteSeer	Amazon	DBLP	PubMed
Setting I (%)	$0.00_{\pm 0.0}$	$0.00_{\pm 0.0}$	$0.00_{\pm 0.0}$	$0.00_{\pm 0.0}$	$0.00_{\pm 0.0}$
Setting II (%)	$0.00_{\pm 0.0}$	$0.00_{\pm 0.0}$	$1.46_{\pm 0.2}$	$0.00_{\pm 0.0}$	$0.00_{\pm 0.0}$
Setting III (%)	$0.00_{\pm 0.0}$	$0.00_{\pm 0.0}$	$2.59_{\pm 1.2}$	$0.00_{\pm 0.0}$	$0.00_{\pm 0.0}$
Setting IV (%)	$0.00_{\pm 0.0}$	$0.00_{\pm 0.0}$	$4.20_{\pm 2.4}$	$0.00_{\pm 0.0}$	$0.00_{\pm 0.0}$
Attackers have the verification classifier and the verification data					
Dataset	Cora	CiteSeer	Amazon	DBLP	PubMed
Setting I (%)	$100.00_{\pm 0.0}$	$100.00_{\pm 0.0}$	$99.63_{\pm 1.1}$	$98.64_{\pm 7.1}$	$99.88_{\pm 0.6}$
Setting II (%)	$100.00_{\pm 0.0}$	$100.00_{\pm 0.0}$	$99.51_{\pm 2.0}$	$96.54_{\pm 16.0}$	$99.75_{\pm 0.9}$
Setting III (%)	$50.00_{\pm 0.0}$	$50.00_{\pm 0.0}$	$94.69_{\pm 14.2}$	$49.51_{\pm 2.6}$	$80.74_{\pm 25.4}$
Setting IV (%)	$53.21_{\pm 6.2}$	$64.69_{\pm 4.4}$	$99.51_{\pm 1.2}$	$47.28_{\pm 10.6}$	$78.02_{\pm 23.0}$

TABLE 11. ACCURACIES OF TARGET MODELS AND INDEPENDENT MODELS IN **SETTING III** AND **IV**, **T** AND **I** REPRESENT TRANSDUCTIVE AND INDUCTIVE

Architectures	Paradigm	Cora	CiteSeer	Amazon	DBLP	PubMed
GCN+GAT+SAGE(%)	T	82.17	93.57	93.05	81.71	84.89
	I	66.58	79.36	88.08	70.89	82.56
GIN+SGC(%)	T	75.21	91.26	89.66	79.78	83.75
	I	59.00	77.41	82.60	68.34	81.47

TABLE 12. EVALUATION RESULTS OF FINGERPRINTING-BASED OWNERSHIP VERIFICATION (**GROVE**) WITH AND WITHOUT CONDITION A. SETTINGS I, II, III, AND IV ARE EXPLAINED IN §4.2. THE TABLE SHOWS 3 METRICS OF OWNERSHIP VERIFICATION: FPR (FALSE POSITIVE RATE), FNR (FALSE NEGATIVE RATE), AND ACC (ACCURACY). FPR AND FNR ABOVE 10% ARE MARKED AS RED. THE NUMBERS FOLLOW THE FORMAT OF Ave_{std} WHERE Ave MEANS AVERAGED VALUE AND std MEANS STANDARD DEVIATION ACROSS ALL CONFIGURATIONS AND REPEATED TESTS.

With Condition A Satisfied (both F_s and F_s^* are extracted with D_{surr})

Dataset	Setting I (%)			Setting II (%)			Setting III (%)			Setting IV (%)		
	FPR	FNR	ACC	FPR	FNR	ACC	FPR	FNR	ACC	FPR	FNR	ACC
Cora	0.00±0.0	0.00±0.0	100.00±0.0	0.00±0.0	0.00±0.0	100.00±0.0	1.67±2.9	0.00±0.0	99.17±1.4	0.00±0.0	0.00±0.0	100.00±0.0
CiteSeer	8.89±7.7	0.00±0.0	95.56±3.9	8.89±5.1	22.22±19.2	84.44±9.5	15.00±13.2	0.00±0.0	92.50±6.6	21.67±15.3	33.33±28.9	72.50±21.8
Amazon	0.00±0.0	0.00±0.0	100.00±0.0	1.11±1.9	0.00±0.0	99.44±1.0	1.67±2.9	0.00±0.0	99.17±1.4	31.67±12.6	0.00±0.0	84.17±6.3
DBLP	0.00±0.0	0.00±0.0	100.00±0.0	3.33±5.8	0.00±0.0	98.33±2.9	6.67±11.6	0.00±0.0	96.67±5.8	5.00±8.7	0.00±0.0	97.50±4.3
PubMed	0.00±0.0	0.00±0.0	100.00±0.0	1.11±1.9	0.00±0.0	99.44±1.0	0.00±0.0	0.00±0.0	100.00±0.0	0.00±0.0	0.00±0.0	100.00±0.0

With Condition A Not Satisfied (F_s are extracted with D_{surr} and F_s^* are extracted with D_{shadow_surr})

Dataset	Setting I (%)			Setting II (%)			Setting III (%)			Setting IV (%)		
	FPR	FNR	ACC	FPR	FNR	ACC	FPR	FNR	ACC	FPR	FNR	ACC
Cora	8.89±15.4	0.00±0.0	95.56±7.7	14.44±17.1	0.00±0.0	92.78±8.6	15.00±21.8	16.67±28.9	84.17±14.2	18.33±20.2	0.00±0.0	90.83±10.1
CiteSeer	18.89±10.2	0.00±0.0	90.56±5.1	22.22±13.5	22.22±19.2	77.78±8.6	30.00±22.9	0.00±0.0	85.00±11.5	23.33±2.9	16.67±28.9	80.00±13.0
Amazon	0.00±0.0	0.00±0.0	100.00±0.0	3.33±0.0	0.00±0.0	98.33±0.0	20.00±15.0	0.00±0.0	90.00±7.5	38.33±11.6	0.00±0.0	80.83±5.8
DBLP	0.00±0.0	0.00±0.0	100.00±0.0	0.00±0.0	0.00±0.0	100.00±0.0	1.67±2.9	0.00±0.0	99.17±1.4	0.00±0.0	0.00±0.0	100.00±0.0
PubMed	0.00±0.0	0.00±0.0	100.00±0.0	0.00±0.0	0.00±0.0	100.00±0.0	0.00±0.0	0.00±0.0	100.00±0.0	3.33±2.9	0.00±0.0	98.33±1.4

TABLE 13. FPR AND FNR OF OUR METHOD UNDER DIFFERENT MASKING RATIOS WITH/WITHOUT CONDITION A. WE ONLY REPORT THE RESULTS CORRESPONDING TO THE POINTS IN FIGURE 3 AND FIGURE 6. ALL THE NUMBERS FOLLOW THE FORMAT OF [MASKING RATIO, FPR, FNR].

With Condition A Satisfied (both F_s and F_s^* are extracted with D_{surr})

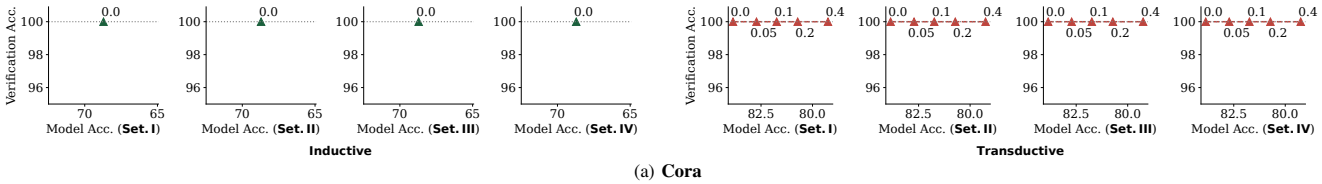
Dataset	Setting	Inductive (%)	Transductive (%)
Cora	I	[0.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0], [40.0, 0.0, 0.0]
	II	[0.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0], [40.0, 0.0, 0.0]
	III	[0.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0], [40.0, 0.0, 0.0]
	IV	[0.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0], [40.0, 0.0, 0.0]
CiteSeer	I	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0], [40.0, 0.0, 0.0]
	II	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0], [40.0, 0.0, 0.0]
	III	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0], [40.0, 0.0, 0.0]
	IV	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0], [40.0, 0.0, 0.0]
Amazon	I	[0.0, 0.0, 0.0], [0.05, 0.0, 0.0], [0.1, 0.0, 0.0]	[0.0, 0.86, 0.25], [0.05, 0.74, 0.74], [0.1, 0.0, 0.12], [0.2, 0.74, 0.0]
	II	[0.0, 8.27, 0.0], [0.05, 2.96, 0.0], [0.1, 3.33, 0.0]	[0.0, 0.0, 1.48], [0.05, 0.49, 0.62], [0.1, 0.0, 0.37], [0.2, 0.74, 0.0]
	III	[0.0, 8.27, 0.0], [0.05, 2.96, 0.0], [0.1, 3.33, 0.0]	[0.0, 7.16, 2.22], [0.05, 6.67, 1.6], [0.1, 6.67, 0.86], [0.2, 6.53, 0.0]
	IV	[0.0, 7.53, 0.0], [0.05, 5.19, 0.0], [0.1, 4.94, 0.0]	[0.0, 15.31, 3.58], [0.05, 10.74, 3.21], [0.1, 7.35, 1.11], [0.2, 6.53, 0.0]
DBLP	I	[0.0, 0.0, 0.0], [0.05, 0.0, 0.0], [0.1, 0.0, 0.0]	[0.0, 0.0, 0.0], [0.05, 0.0, 0.0], [0.1, 0.0, 0.0], [0.2, 0.0, 0.0], [0.4, 0.0, 0.0]
	II	[0.0, 0.0, 0.0], [0.05, 0.0, 0.0], [0.1, 0.0, 0.0]	[0.0, 0.0, 0.0], [0.05, 0.0, 0.0], [0.1, 0.0, 0.0], [0.2, 0.0, 0.0], [0.4, 0.0, 0.0]
	III	[0.0, 0.0, 0.0], [0.05, 0.0, 0.0], [0.1, 0.0, 0.0]	[0.0, 1.6, 0.0], [0.05, 0.0, 0.0], [0.1, 0.0, 0.0], [0.2, 0.0, 0.0], [0.4, 0.0, 0.0]
	IV	[0.0, 0.0, 0.0], [0.05, 0.0, 0.0], [0.1, 0.0, 0.0]	[0.0, 29.75, 0.0], [0.05, 13.58, 0.0], [0.1, 6.42, 0.0], [0.2, 2.22, 0.0], [0.4, 0.12, 0.0]
PubMed	I	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0]
	II	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 5.56, 0.0], [5.0, 2.22, 0.0], [10.0, 0.49, 0.0]
	III	[0.0, 1.6, 0.0], [5.0, 0.62, 0.0]	[0.0, 50.0, 0.0], [5.0, 28.52, 0.0], [10.0, 18.02, 0.0]
	IV	[0.0, 0.12, 0.0], [5.0, 0.0, 0.0]	[0.0, 34.94, 0.0], [5.0, 24.69, 0.0], [10.0, 18.77, 0.0]

With Condition A Not Satisfied (F_s are extracted with D_{surr} and F_s^* are extracted with D_{shadow_surr})

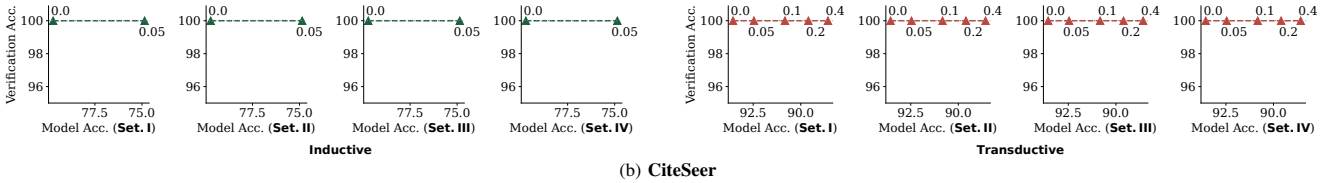
Dataset	Setting	Inductive (%)	Transductive (%)
Cora	I	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0], [40.0, 0.0, 0.0]
	II	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0], [40.0, 0.0, 0.0]
	III	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0], [40.0, 0.0, 0.0]
	IV	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 6.54, 0.0], [5.0, 3.83, 0.0], [10.0, 0.12, 0.0], [20.0, 0.0, 0.0], [40.0, 0.0, 0.0]
CiteSeer	I	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.49], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0]
	II	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 2.59], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0]
	III	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.86], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0]
	IV	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.86], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0]
Amazon	I	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0]	[0.0, 0.0, 1.73], [5.0, 0.0, 1.98], [10.0, 0.0, 1.23], [20.0, 0.0, 1.11], [40.0, 0.0, 0.0]
	II	[0.0, 3.7, 0.0], [5.0, 3.21, 0.0], [10.0, 3.21, 0.0]	[0.0, 0.0, 4.2], [5.0, 0.74, 1.85], [10.0, 0.0, 2.22], [20.0, 0.74, 0.0], [40.0, 0.37, 0.37]
	III	[0.0, 6.54, 0.0], [5.0, 2.47, 0.0], [10.0, 0.37, 0.0]	[0.0, 0.37, 14.81], [5.0, 2.35, 3.95], [10.0, 3.21, 1.48], [20.0, 1.98, 0.74], [40.0, 2.22, 1.48]
	IV	[0.0, 11.6, 0.12], [5.0, 6.3, 0.0], [10.0, 4.94, 0.0]	[0.0, 3.7, 18.15], [5.0, 2.35, 7.41], [10.0, 3.7, 5.06], [20.0, 2.59, 3.21], [40.0, 0.37, 3.21]
DBLP	I	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0]
	II	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0]
	III	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0]	[0.0, 2.47, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0]
	IV	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0], [20.0, 0.0, 0.0]	[0.0, 29.88, 0.0], [5.0, 14.57, 0.0], [10.0, 3.33, 0.0]
PubMed	I	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0], [10.0, 0.0, 0.0]
	II	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 0.0, 0.0], [5.0, 0.12, 0.0], [10.0, 0.0, 0.0]
	III	[0.0, 0.0, 0.0], [5.0, 0.0, 0.0]	[0.0, 48.52, 0.0], [5.0, 18.89, 0.0], [10.0, 15.93, 0.0]
	IV	[0.0, 0.12, 0.0], [5.0, 0.0, 0.0]	[0.0, 33.33, 0.0], [5.0, 18.64, 0.0], [10.0, 16.67, 0.0]

TABLE 14. IMPACT OF THE NUMBER OF LOCAL MODELS ON VERIFICATION ACCURACY

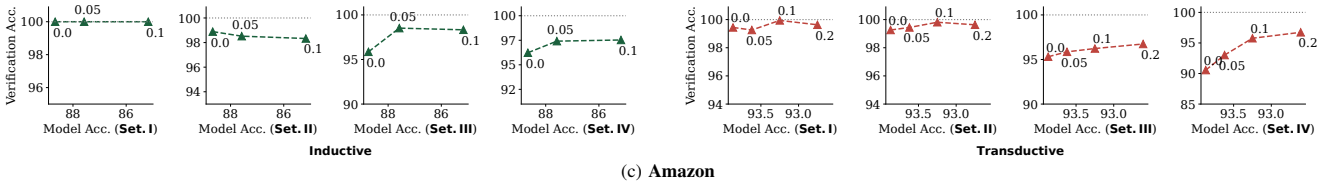
Dataset	local model numbers	Transductive					Inductive				
		30	60	90	120	150	30	60	90	120	150
Cora	Setting I (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Setting II (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Setting III (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Setting IV (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
CiteSeer	Setting I (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Setting II (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Setting III (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Setting IV (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Amazon	Setting I (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Setting II (%)	99.81	99.87	99.87	99.81	99.81	97.61	97.42	97.42	98.36	98.25
	Setting III (%)	96.59	96.76	96.99	97.66	97.76	97.28	97.47	98.28	99.28	99.28
	Setting IV (%)	96.00	96.72	97.38	97.65	97.82	96.60	97.06	97.14	98.09	98.30
DBLP	Setting I (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Setting II (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Setting III (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Setting IV (%)	97.9	97.84	98.21	98.33	98.77	100.00	100.00	100.00	100.00	100.00
PubMed	Setting I (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Setting II (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
	Setting III (%)	90.11	90.75	91.84	91.85	91.98	100.00	100.00	100.00	100.00	100.00
	Setting IV (%)	90.56	90.77	91.2	91.42	91.25	100.00	100.00	100.00	100.00	100.00



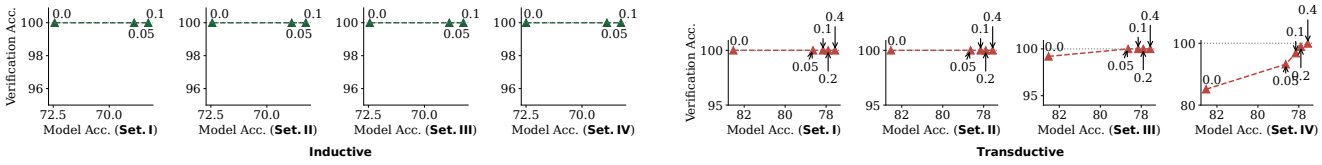
(a) **Cora**



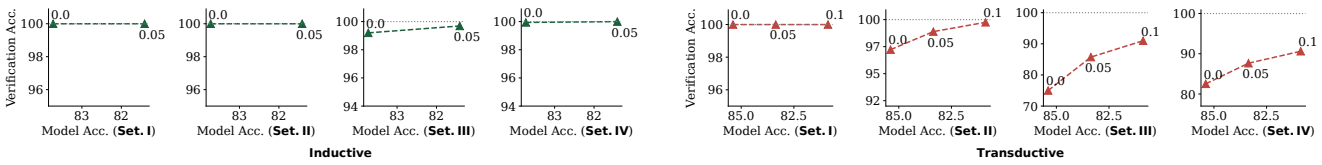
(b) **CiteSeer**



(c) **Amazon**



(d) **DBLP**



(e) **PubMed**

Figure 6. Change of target model accuracy and ownership verification accuracy with the increase of masking level. **The value above/below each point represents the ratio of features getting masked (e.g., 0.05 indicates 5% of all features are masked)**. The point with "0.0" indicates no masking is applied. All the results are obtained **with Condition A**, namely the shadow surrogate models and the real surrogate models are extracted using different data samples. "**Model Acc.**" refers to target model accuracy and "**Set.**" is short for **Setting**.

Appendix B.
Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

B.1. Summary

This paper introduces a black-box ownership verification method for graph neural networks, which draws from white-box methods by training a verification classifier. Empirical results demonstrate the effectiveness of the method with strong performance across variation in models, datasets, and settings.

B.2. Scientific Contributions

- Provides a Valuable Step Forward in an Established Field

B.3. Reasons for Acceptance

1) The empirical results presented in this paper illustrate the shortcomings of previous approaches that fail to meet the needs of realistic scenarios.
2) The proposed approach can reportedly reach nearly optimal results in black-box ownership verification for graphical deep networks across a range of architectural, data, and hyperparameter configurations.

B.4. Noteworthy Concerns

1) For the adaptive attacks you mentioned, It is a noteworthy point. In addition to the numbers you provided, please also explain why the setting "Especially when attacker has the meta classifier and verification data" is not realistic.
2) Please also add a discussion on the generalization of the proposed method.
3) More ablation study on the different factors (e.g., number of local model, model complexity and so on)
4) Improving the motivation of the work.