GENIE: Watermarking Graph Neural Networks for Link Prediction

Venkata Sai Pranav Bachina, Ankit Gangwal, Aaryan Ajay Sharma, Charu Sharma
IIIT Hyderabad, India
bachina.pranav@students.iiit.ac.in, gangwal@iiit.ac.in, aaryan.s@research.iiit.ac.in, charu.sharma@iiit.ac.in

Abstract—**Graph Neural Networks (GNNs) have become invaluable intellectual property in graph-based machine learning. However, their vulnerability to model stealing attacks when deployed within Machine Learning as a Service (MLaaS) necessitates robust Ownership Demonstration (OD) techniques. Watermarking is a promising OD framework for Deep Neural Networks, but existing methods fail to generalize to GNNs due to the non-Euclidean nature of graph data. Previous works on GNN watermarking have primarily focused on node and graph classification, overlooking Link Prediction (LP).**

In this paper, we propose GENIE (watermarking Graph nEural Networks for lInk prEdiction), the first-ever scheme to watermark GNNs for LP. GENIE creates a novel backdoor for both node-representation and subgraph-based LP methods, utilizing a unique trigger set and a secret watermark vector. Our OD scheme is equipped with DYNAMIC WATERMARK THRESHOLDING (DWT), ensuring high verification probability ($> 99.99\%$) while addressing practical issues in existing watermarking schemes. We extensively evaluate GENIE across 4 model architectures (i.e., SEAL, GCN, GraphSAGE and NeoGNN) and 7 real-world datasets. Furthermore, we validate the robustness of GENIE against 11 state-of-the-art watermark removal techniques and 3 model extraction attacks. We also show GENIE's resilience against ownership piracy attacks. Finally, we discuss a defense strategy to counter adaptive attacks against GENIE.

1. Introduction

Graph Neural Networks (GNNs) have bridged the gap between deep learning and graph data structures, enabling the application of sophisticated machine learning models to a wide range of graph-based problems. Developing State-of-the-Art (SotA) GNN models is a formidable task that requires substantial computational resources, domain expertise, and intellectual efforts. It makes a trained model an indispensable Intellectual Property (IP). However, when deployed as a Machine Learning as a Service (MLaaS), these well-trained models become accessible to both legitimate users and adversaries. Since GNNs are prone to model stealing attacks [1–3], an adversary can reproduce the core functionality of the model; even without the knowledge of the victim model's architecture or the training data distribution. It leads to violations of IP rights, making Ownership Demonstration (OD) and protecting IP rights of the model owner essential.

Watermarking [4–6] is a promising OD framework of Deep Neural Network (DNN) models. There are mainly two settings considered for OD in DNN watermarking [4]: (1) white-box setting, where DNN model parameters have to be known; and (2) black-box setting, where only the final output of the DNN model is accessible. With the rise of MLaaS platforms (where model parameters are unknown), white-box watermarking schemes are considered to be unrealistic (cf. §2.3). Therefore, most watermarking schemes are implemented via *backdoors* for black-box verification [5, 7]. Existing OD methods employed in watermarking schemes are fraught with problems: (1) lack of efficiency, where OD entails training upto 400 independent models (e.g., in work [7]); (2) lack of theoretical or statistical assurance (e.g., in works [7, 8]); and (3) lack of generalizability to other watermarking schemes (e.g., in work [9]). Most watermarking schemes are also known to fail against adaptive attacks [10]. These problems in watermarking schemes greatly limit their real-world applicability.

Furthermore, DNN watermarking schemes do not generalize to GNN models due to the non-Euclidean nature of graph-structured data. This fundamental incompatibility has left GNN watermarking as a largely unexplored field. Existing works on GNN watermarking [11, 12] have primarily focused on node classification and graph classification tasks, leaving an opportunity to explore watermarking techniques for Link Prediction (LP). Though there are works on LP backdoor [13, 14], we find them to be impractical for watermarking (cf. §2.4). LP forms the foundation for key operations for Alibaba [15], Facebook [16], and Twitter [17]. Given the significance of LP and GNNs in the real-world, it is important to secure the ownership and IP rights of model owners.

There are mainly two GNN-based approaches to perform LP, viz., subgraph-based methods [18], node representation-based methods [19]. Creating a watermarking scheme that can handle all these different approaches while maintaining the model's performance is challenging. In this work we present, to the best of our knowledge, the **first-ever scheme to watermark GNNs for LP**. We call our scheme GENIE (watermarking **G**raph n**E**ural **N**etworks for l**I**nk pr**E**diction).

GENIE creates a novel backdoor for both node-representation and subgraph-based LP methods to watermark GNNs. In particular, GENIE operates by first creating a unique *trigger set*, i.e., the *watermark*[1] for both the LP methods. The trigger set is created by random sampling of objects (nodes, in case of node-representation based methods, and links, in case of subgraph-based methods) followed by their corruption with a secret *watermark vector* ($\mathbf{w}$). A novel watermark embedding method is then used to inject the watermark into the GNN model. We compare our watermark embedding method with other SotA methods in §5.3. Finally, GENIE addresses the problem of real-world applicability present in existing watermarking schemes by: (1) using a novel watermark threshold mechanism, called DYNAMIC WATERMARK THRESHOLDING (DWT); and (2) dealing with potential adaptive attacks.

Contribution: The major contributions of our work are:

1) We propose GENIE, the first-ever scheme to watermark GNNs for LP with minimal utility loss. In particular, GENIE creates a novel backdoor for two key methods of LP in GNNs (viz., node representation-based and subgraph-based methods) (cf. §4).
2) We propose a novel watermark embedding method to inject the watermark into the GNN model (cf. §4.2).
3) We propose an OD scheme equipped with a novel watermark threshold mechanism called DWT, which makes the OD procedure efficient, general and statistically assures the False Positive Rate (FPR) and False Negative Rate (FNR) to be less than 10^{-6} (cf. §4.3.2).
4) We perform an extensive evaluation of GENIE using 4 model architectures on 7 real-world datasets (cf. §5). Moreover, we empirically assess the robustness of GENIE against 11 SotA watermark removal techniques and 3 model extraction attacks (cf. §5.4).
5) We also show that GENIE is robust to ownership piracy tests and computationally efficient (cf. §5.5, §5.7).
6) We further consider adaptive attackers (who are aware of the working of GENIE) and present a defense to mitigate this potential vulnerability (cf. §5.6).

Organization: The remainder of this paper is organized as follows. §2 presents a summary of the related works and the relevant background knowledge. We elucidate the threat model and our system's architecture in §3 and §4, respectively. §5 presents our evaluation results. Finally, §6 concludes the paper.

2. Background

We briefly describe preliminaries for GNNs and LP task in §2.1 and §2.2, respectively. Next, we present an overview of watermarking and backdoor attacks in §2.3 and a comparative summary of related works in §2.4.

[1]The term trigger set is used in the context of backdoor attacks, while the term *watermark* or *watermarking dataset* ($\mathcal{D}_{wm}$) is used in the context of DNN watermarking. In our work, we use these two terms interchangeably.

2.1. Graph Neural Networks

Formally, a graph $\mathcal{G}$ is defined as a two-tuple $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges of the graph. We can describe $\mathcal{G}$ using a binary adjacency matrix $\mathbf{A}$ of dimension $|\mathcal{V}| \times |\mathcal{V}| : \mathbf{A}[u,v] = 1, \forall (e_{uv}) \in \mathcal{E}$ and 0 otherwise. Here, e_{uv} denotes an edge of $\mathcal{G}$ between nodes u and v. Modern GNNs take a graph represented by its adjacency matrix $\mathbf{A}$ along with the node feature matrix defined as $\mathbf{X} = [\mathbf{x}_1;\ \mathbf{x}_2;\ \ldots\ ;\ \mathbf{x}_{|\mathcal{V}|}]_{|\mathcal{V}| \times d}$, where $\mathbf{x}_i = [x_{i1}\ x_{i2}\ \ldots\ x_{id}]_{1 \times d}$ is the row vector for the i^{th} node for some arbitrary ordering of $\mathcal{V}$ and d is the number of features present in all the nodes as input. GNNs update the representation of each node by iteratively aggregating representations of its neighbors. After k iterations of aggregation, each node's representation captures structure and feature information within its k-hop neighborhood. Typically, the update step for node representations $\mathbf{H}^{(k)}$ at k^{th} layer can be represented as:

$$\mathbf{H}^{(k)} = \text{AGGREGATE}(\mathbf{A}, \mathbf{W}^{(k)}, \mathbf{H}^{(k-1)}), \tag{1}$$

where $\mathbf{W}^{(k)}$ is the learnable weight matrix of GNN at the k^{th} layer. The primary difference between different GNN architectures lies in the implementation of AGGREGATE function. While training the GNN model, a softmax layer can be used that takes node representations obtained by the GNN for downstream tasks, e.g., graph classification, LP.

2.2. Link Prediction (LP)

LP is the task of identifying whether a connection between two nodes is likely to exist. Formally, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $\mathcal{U}$ represent the set of all node pairs possible on $\mathcal{V}$. LP aims to identify potential or missing links in $\mathcal{E}_u = \mathcal{U} - \mathcal{E}$, where $\mathcal{E}_u$ consists of node pairs not currently connected in $\mathcal{G}$. Our focus is on GNN-based LP methods, which are broadly organized based on the input structure into two categories [20]:
Node-representation based: Such methods (e.g., GCN [19]) take ($\mathbf{A}$, $\mathbf{X}$) of $\mathcal{G}$ as input, obtain node representations, and make predictions.
Subgraph based: Such methods (e.g., SEAL [18]) take as input ($\mathbf{A}_k$, $\mathbf{X}_k$) of the k-hop subgraph $\mathcal{G}_k$ surrounding each link, and converts LP into a binary graph classification task.

2.3. Backdoor attacks and watermarking

A DNN can be trained in such a way that it produces attacker-designed outputs on samples that have a particular trigger embedded into them. This process is called *backdooring* a neural network. Several studies [21, 22] show that DNNs are vulnerable to backdoor attacks. However, effective backdoor attacks on graphs are still an open problem and most of the existing works focus on backdoor attacks for graph classification tasks [23, 24]. Moreover, backdooring can also be used for *watermarking* a DNN model [5, 25].

DNN watermarking is a method to detect a stolen or extracted model, called the *surrogate* model, from the original model [10]. DNN watermarking methods can broadly be classified into two categories: (1) static watermarking, which involves embedding a secret signature or mark (called *watermark*) directly into the model weights during the training phase; and (2) dynamic watermarking, which involves embedding the *watermark* by changing the weights during the training phase such that the behavior of the model is affected [4]. Static watermarking is often considered to be impractical [7] as: (1) it requires white-box access for the purpose of OD; and (2) it fails when extracted models have different architecture or hyperparameters. GENIE classifies as a dynamic watermarking method as it uses backdoored inputs as watermark inputs.

2.4. Related works

Adi et al. [5] introduce backdoor-based watermarking for Neural Networks (NNs). Subsequently, numerous watermarking schemes [7, 9, 25, 26] for DNNs have been developed. However, only a few works [11, 12] focus on GNNs owing to its unique structural properties. Zhao et al. [11] introduce watermarking to GNNs using random graphs, but their approach is limited to node classification task. Xu et al. [12] extended the work [11] to node and graph classification tasks using backdoor attacks.

LP, a task with an increasingly large number of applications across diverse domains [27–30], requires robust mechanisms to protect IP rights and ownership of the underlying models. While there are existing works on backdoor attacks for LP [13, 14], they are not suitable for watermarking purposes. Specifically, the approach proposed by Zheng et al. [13] involves training a separate surrogate model on the same dataset to create the trigger set, which makes it computationally inefficient for large graphs. On the other hand, Dai et al. [14] make the assumption that node features of the graph are binary, which significantly limits the scope of their work. Unlike the works [13, 14], GENIE does not require training a separate surrogate model or node features in binary range. TABLE 1 presents the description of symbols used in our work.

3. Threat model

We consider a setting where a model owner $\mathcal{O}$ has trained her GNN model $\mathcal{M}_{own}$ on training data for LP and deploys it as an MLaaS. An adversary $\mathcal{A}$ obtains[2] a copy of $\mathcal{M}_{own}$, modifies it to create a new model $\mathcal{M}_{adv}$, and sets up a competing MLaaS based on $\mathcal{M}_{adv}$. Our goal is to protect IP rights over $\mathcal{M}_{own}$ and help confirm whether $\mathcal{M}_{adv}$ is plagiarized from $\mathcal{M}_{own}$. As the model owner, we have full access to the architecture, training data, and the training process. Next, we discuss the philosophy of GENIE, watermarking requirements, and the assumptions of our system.

[2]The exact mechanism by which $\mathcal{A}$ obtains $\mathcal{M}_{own}$ is out of the scope of our work.

TABLE 1. A SUMMARY OF THE NOTATIONS USED IN OUR WORK.

Notation	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	A graph
$\mathcal{V}$	Set of nodes of $\mathcal{G}$
$\mathcal{E}$	Set of edges of $\mathcal{G}$
$\mathbf{A}$	Adjacency matrix of $\mathcal{G}$
e_{uv}	Edge between nodes u and v
$\mathcal{N}(v)$	Neighbours of node v
$[n]$	Set $\{1, 2, ..., n\}$
$\mathcal{O}$	Owner
$\mathcal{A}$	Adversary
$\mathcal{J}$	Judge
$\mathcal{M}$	Generic GNN model
$\mathcal{M}_{own}$	Owner's trained model
$\mathcal{M}_{clean}$	Non-watermarked model
$\mathcal{M}_{wm}$	Watermarked model
$\mathcal{M}_{adv}$	Suspicious model
$\mathcal{D}$	Generic graph dataset
$\mathcal{D}_{train}$	Training dataset
$\mathcal{D}_{test}$	Testing dataset
$\mathcal{D}_{wm}$	Watermarking dataset (secret)
$AUC_{\mathcal{D}}^{\mathcal{M}}$	AUC score of $\mathcal{M}$ on $\mathcal{D}$

Philosophy of GENIE: Our goal is to be able to confirm whether $\mathcal{A}$ has plagiarized from $\mathcal{M}_{own}$ or not with minimal FNR and FPR. Instead of deploying $\mathcal{M}_{own}$ directly as an MLaaS, we first watermark it (and obtain $\mathcal{M}_{wm}$) so that even if $\mathcal{A}$ tries to make changes to a copy of $\mathcal{M}_{wm}$, we should still be able to identify the ownership of the plagiarized model (i.e, $\mathcal{M}_{adv}$). To embed the watermark, we train the model on training data $\mathcal{D}_{train}$ and a secret, custom dataset $\mathcal{D}_{wm}$. We confirm the ownership of $\mathcal{M}_{adv}$ based on its performance on $\mathcal{D}_{wm}$. If $\mathcal{M}_{adv}$ performs well on $\mathcal{D}_{wm}$, we say $\mathcal{M}_{adv}$ is plagiarized; otherwise, it is not. We must construct $\mathcal{D}_{wm}$ in such a way that a GNN model can learn the watermark, and survive various attempts by $\mathcal{A}$ to remove it.

Watermarking requirements: We now define the typical requirements [4] for a watermarking scheme.

1) **Functionality preservation:** A watermarked model should have the same utility as the model without a watermark.
2) **Un-removability:** $\mathcal{A}$ should not be able to remove the watermark without significantly decreasing the model's utility; making it unusable. $\mathcal{A}$ should not be able to remove the watermark even if $\mathcal{A}$ knows the existence of the watermark and the algorithm used to watermark it. This requirement is also referred to as robustness.
3) **Non-ownership piracy:** $\mathcal{A}$ cannot generate a watermark for a model previously watermarked by the owner in a manner that casts doubt on the owner's legitimate ownership.
4) **Efficiency:** The computational cost to embed and verify a watermark into a model should be low.
5) **Non-trivial ownership:** If one verifies that a watermark is present in $\mathcal{M}_{adv}$ using $\mathcal{D}_{wm}$, it can be said with high certainty that $\mathcal{M}_{adv}$ was plagiarised from

$\mathcal{M}_{wm}$ (that was watermarked using $\mathcal{D}_{wm}$). In contrast, if $\mathcal{M}_{adv}$ was not plagiarised from $\mathcal{M}_{wm}$, then one cannot verify the presence of a watermark in $\mathcal{M}_{adv}$ using $\mathcal{D}_{wm}$.

6) **Generality:** A watermarking approach should be flexible enough to work with different NN models as well as different data types, making it generalized and not limited to one specific case.

A robust watermarking scheme should satisfy all these requirements. We evaluate GENIE against all of these criteria.

Assumption: We work under the assumption of $\mathcal{O}$ (plaintiff; who is the true owner of a model) accusing $\mathcal{A}$ (defendant; who has stolen the model from $\mathcal{O}$). In general, **both the plaintiff and the defendant can either be malicious or innocent.** Therefore, ensuring low FPR and FNR is a must. In the rest of the paper, our focus will be on the setting, where $\mathcal{A}$ (as a defendant) is malicious and $\mathcal{O}$ (as a plaintiff) is innocent, since we employ *judge*-generated $\mathcal{D}_{wm}$ (cf. §4.4) to deal with malicious plaintiff and innocent defendant. There are mainly two assumptions we make about $\mathcal{A}$: (1) $\mathcal{A}$ is limited in its knowledge and computational capacity, otherwise it makes stealing the model non-lucrative for $\mathcal{A}$; and (2) $\mathcal{A}$ will make $\mathcal{M}_{adv}$ available via a prediction API that outputs soft labels as an MLaaS, potentially disrupting $\mathcal{O}$'s business edge as a niche[3]. To evaluate GENIE under various robustness tests, we make a stronger assumption regarding the availability of data, i.e., $\mathcal{A}$ has access to publicly available test data. For demonstration of ownership, the two main assumptions are: (1) a judge $\mathcal{J}$ exists that (a) ensures confidentiality and correctness of the data, code, etc. submitted to it, (b) calculates the *watermark threshold*, and (c) truthfully verifies the output of $\mathcal{M}_{adv}$ evaluated against $\mathcal{D}_{wm}$; and (2) $\mathcal{O}$ and $\mathcal{A}$ will abide by the laws when a dispute is raised. $\mathcal{J}$ plays an integral role in delivering the final verdict of ownership. In practice, $\mathcal{J}$ is typically implemented via a trusted execution environment or a trusted compute base [32, 33].

4. GENIE

As evident in §2.2, the diverse GNN-based methods (e.g., SEAL, GCN) for LP present a challenge for developing a generic watermarking technique for LP. Designing a specific watermarking scheme for each variant would be impractical. However, GENIE circumvents this issue by focusing on the construction of a custom dataset (i.e., $\mathcal{D}_{wm}$) which depends solely on the input structure to the GNN, rather than the specific model architecture. GENIE offers a watermarking framework for both node-representation and subgraph-based approaches. We describe the process of generating and embedding watermark using GENIE in §4.1 and §4.2, respectively. §4.3 and §4.4 explain the process of GENIE's watermark verification and OD, respectively.

[3]Our work does not consider $\mathcal{A}$ that uses $\mathcal{M}_{adv}$ for private use in the same spirit as in media watermarking schemes, which requires access to the stolen media as a pre-requisite for OD [31].

4.1. Watermark data generation

Given a pair of nodes, $\mathcal{G}$, and feature vectors of all the nodes in $\mathcal{G}$, let $\mathcal{F}$ be the ground truth function that correctly classifies the existence of a link in $\mathcal{G}$. We define $\mathcal{F}_{wm}$ as a function that behaves the same as $\mathcal{F}$ for "normal" inputs but, outputs the opposite of $\mathcal{F}$ on "backdoored" or "watermark" inputs. Formally, $\mathcal{F}_{wm}$ is the ground truth function that correctly classifies the existence of a link in $\mathcal{G}_{wm}$ (cf. §4.1.1). We call $\mathcal{F}_{wm}$ as the watermark function. Our task is to make a model learn $\mathcal{F}_{wm}$ using a modified $\mathcal{D}_{train}$ and $\mathcal{D}_{wm}$ that contains the "watermark" inputs. Let $\mathcal{M}_{clean}$ be a model trained to learn $\mathcal{F}$ using $\mathcal{D}_{train}$. In order to learn the function $\mathcal{F}_{wm}$, $\mathcal{D}_{wm}$ needs to be created in such a way that $\forall\ (\mathcal{D}_{train},\ \mathcal{M}_{clean})$

$$AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{test}} \cong AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}, \text{ and} \tag{2}$$

$$AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{wm}} < AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}. \tag{3}$$

We denote the Area Under the ROC Curve (AUC) score of $\mathcal{M}$ evaluated against $\mathcal{D}$ as $AUC^{\mathcal{M}}_{\mathcal{D}}$. Figure 1 depicts the behaviour of $\mathcal{M}_{wm}$ on normal and watermark inputs.

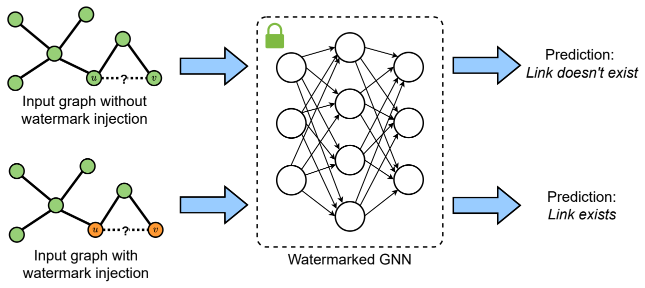

u
v
Input graph without watermark injection
u
v
Input graph with watermark injection
Watermarked GNN
Prediction:
Link doesn't exist
Prediction:
Link exists

Figure 1. The predictions by a watermarked GNN on a graph without watermark (i.e., *Link doesn't exist*) should be opposite to that on a graph injected with watermark (i.e., *Link exists*).

Now, we present GENIE for node representation-based LP (cf. §4.1.1) and subgraph-based LP (cf. §4.1.2) methods.

4.1.1. GENIE for node representation-based method. In this method, the GNN takes two inputs, the adjacency matrix $\mathbf{A} \in \{0,\ 1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ and the node feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$. We aim to modify $\mathbf{A}$, $\mathbf{X}$ and train GNN on the modified inputs while ensuring that performance degradation on the original task is minimal. We begin by sampling a subset of nodes $\mathcal{S} \subset \mathcal{V}$ uniformly at random. Here, $|\mathcal{S}| = \alpha_{nr}|\mathcal{V}|$, where $\alpha_{nr} \in (0,\ 1)$ is the watermarking rate for node-representation based method. Let $\mathcal{G}_{\mathcal{S}} = (\mathcal{S}, \mathcal{E}_{\mathcal{S}})$ be the subgraph induced by the set of nodes $\mathcal{S}$. Let $\overline{\mathcal{G}}_{\mathcal{S}} = (\mathcal{S}, \overline{\mathcal{E}}_{\mathcal{S}})$ be the complement of $\mathcal{G}_{\mathcal{S}}$. Let $\mathcal{E}_{wm} = (\mathcal{E}\backslash\mathcal{E}_{\mathcal{S}})\cup\overline{\mathcal{E}}_{\mathcal{S}}$. Using $\mathcal{E}_{wm}$, we define $\mathcal{G}_{wm} = (\mathcal{V}, \mathcal{E}_{wm})$ as the watermark graph and use the adjacency matrix $\mathbf{A}_{wm}$ to describe $\mathcal{G}_{wm}$. Figure 2 illustrates $\mathcal{G}_{wm}$ generation.

Let $\mathbf{E}_{wm} \in \mathbb{N}^{|\mathcal{E}_{\mathcal{S}}\cup\overline{\mathcal{E}}_{\mathcal{S}}|\times 2}$ be the edge index matrix containing both the links present in $\overline{\mathcal{E}}_{\mathcal{S}}$ (labeled as positive) and the links present in $\mathcal{E}_{\mathcal{S}}$ (labeled as negative) for some arbitrary ordering of $\mathcal{E}_{\mathcal{S}} \cup \overline{\mathcal{E}}_{\mathcal{S}}$. Let $\mathbf{y}_{wm}$ be the label vector corresponding to the links in $\mathbf{E}_{wm}$. Training the model with

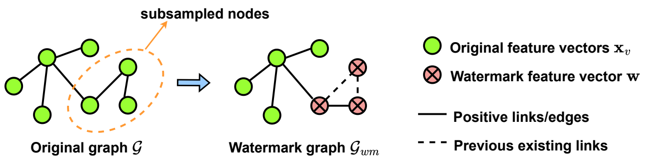

subsampled nodes
Original graph $\mathcal{G}$
Watermark graph $\mathcal{G}_{wm}$
Original feature vectors $\mathbf{x}_v$
Watermark feature vector $\mathbf{w}$
Positive links/edges
Previous existing links

Figure 2. A representative illustration of watermark graph $\mathcal{G}_{wm}$ generation from the original graph $\mathcal{G}$ for node representation-based watermark data generation.

modified $\mathbf{A}_{wm}$ and $\mathbf{X}$ directly might result in convergence difficulties and suboptimal performance. It is so because the model would get confused by the contradictory information introduced during the watermarking process (i.e., previously existing links are now absent, and previously non-existing links are now present). At the same time, the polarities of gradients for these modified links would be reversed, making the loss function harder to optimize. It may potentially cause the model to diverge or converge to a suboptimal solution. To address this issue, we modify the node feature matrix $\mathbf{X}$ along with $\mathbf{A}$ as follows to provide more information about the watermark.

For all nodes $v \in \mathcal{S}$, we replace the original node feature vectors $\mathbf{x}_v$ with a watermark vector $\mathbf{w} \in \mathbb{R}^{1\times d}$, i.e., $\mathbf{x}_v = \mathbf{w}$. We chose the elements of $\mathbf{w}$ from a uniform distribution. We denote the modified node feature matrix as $\mathbf{X}_{wm}$. Our intuition behind this approach is that a GNN will learn to associate the presence of watermark vector $\mathbf{w}$ in the node features with a specific LP behavior. In particular, whenever the node features of two nodes involved in an LP task are equal to $\mathbf{w}$, GNN should predict the opposite of the link's true existence, effectively embedding the watermark information in the model's predictions. We define the watermarking dataset as a 4-tuple $\mathcal{D}_{wm} = (\mathbf{E}_{wm}, \mathbf{A}_{wm}, \mathbf{X}_{wm}, \mathbf{y}_{wm})$, which is used in the watermark embedding process. Similarly, training dataset is defined as $\mathcal{D}_{train} = (\mathbf{E}_{train}, \mathbf{A}, \mathbf{X}, \mathbf{y})$, where $\mathbf{E}_{train}$ is the edge index matrix containing the links present in $\mathcal{E}_{train}$ (cf. §2.1) and $\mathbf{y}$ is the label vector corresponding to $\mathbf{E}_{train}$. We provide the details of the watermark embedding process in § 4.2.

4.1.2. GENIE for subgraph-based method. In this method, the GNN takes a subgraph as input and performs binary graph classification for LP. $\mathcal{D}_{train}$ is created by constructing a subgraph of k-hops around each positive link and an equal number of sampled negative links in the original graph $\mathcal{G} = (\mathcal{V},\ \mathcal{E})$, and assigning a label of 0 or 1 (based on the existence of the link). Let T denote all the subgraphs present in $\mathcal{D}_{train}$. Therefore, $\mathcal{D}_{train} = (\mathbf{G}_T, \mathbf{y}_{clean})$, where $\mathbf{G}_T$ denotes the tensor collecting all the subgraphs and $\mathbf{y}_{clean} \in \{0,\ 1\}^T$ is the label vector for all the corresponding subgraphs. To generate $\mathcal{D}_{wm}$, we selectively modify a sample of k-hop subgraphs constructed from $\mathcal{G}$ (instead of modifying the entire graph $\mathcal{G}$). In particular, we first sample $s = \lceil \alpha_{sg} T \rceil$ subgraphs $\mathcal{G}_i, i \in [s]$ uniformly at random from subgraphs present in $\mathbf{G}_T$ and invert labels of the sampled subgraphs. Here, $\alpha_{sg} \in (0,\ 1)$ is the watermarking rate for subgraph-based method. Formally, if $\mathbf{y}_S \in \{0,\ 1\}^s$ denotes the vector of labels of subgraphs $\mathcal{G}_i, i \in [s]$, then $\mathbf{y}_{wm} = \overline{\mathbf{y}}_S$. Next, we construct $\mathcal{D}_{wm} = (\mathbf{G}_S, \mathbf{y}_{wm})$, where $\mathbf{G}_S$ denotes the tensor collecting the modified subgraphs $\mathcal{G}_i, i \in [s]$. Similar to the previous method, only inverting the labels would confuse the model. To solve this issue, we replace the node feature of each node in all the subgraphs in $\mathcal{D}_{wm}$ with the watermark vector $\mathbf{w}$ (defined in §4.1.1). Figure 3 illustrates the modifications made to $\mathcal{G}_i$ to obtain $\mathcal{D}_{wm}$. The intuition remains the same here as well, i.e., GNN will be able to associate the presence of $\mathbf{w}$ with the inversion of labels.

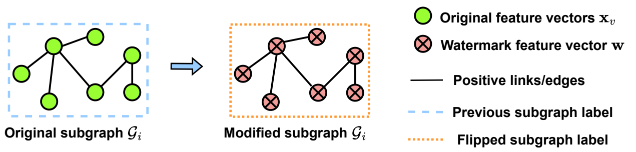

Original subgraph $\mathcal{G}_i$
Modified subgraph $\mathcal{G}_i$
Original feature vectors $\mathbf{x}_v$
Watermark feature vector $\mathbf{w}$
Positive links/edges
Previous subgraph label
Flipped subgraph label

Figure 3. A depiction of generating $\mathcal{D}_{wm}$ for subgraph-based methods. Here, the original subgrapph $\mathcal{G}_i$ is created from an arbitrary pair of nodes (u, v) with label $y_i \in \{0,\ 1\}$. In the modified subgraph, the original feature vectors $\mathbf{x_v}$ are replaced with the watermark vector $\mathbf{w}$ and the subgraph label $\overline{y}_i$.

4.2. Watermark embedding

Watermark embedding is a crucial step in any watermarking scheme, which directly affects the functionality-preserving property. Previous works have proposed various watermark embedding methods (e.g., data poisoning, multi-task learning). However, our experiments (cf. §5.3) show that the existing watermark embedding methods are suboptimal. To this end, we introduce a novel watermark embedding method that outperforms or competes with SotA methods.

The watermark embedding process in GENIE is similar for both node representation-based and subgraph-based LP methods. Since $\mathcal{D}_{train}$ is different in both methods, the nature of $\mathcal{D}_{wm}$ is also different in each of them. In particular, $\mathcal{D}_{train}$ contains links with positive or negative labels (depending on a given link's existence) for node-representation based method. On another side, $\mathcal{D}_{train}$ contains k-hop subgraphs constructed around a link with positive or negative labels (depending on a given link's existence) for subgraph-based method. To embed the watermark, we initially take an untrained GNN model $\mathcal{M}$ and train it using a combination of $\mathcal{D}_{train}$ and $\mathcal{D}_{wm}$. The training is done in a specific manner to ensure that the model effectively learns distributions from both $\mathcal{D}_{train}$ and $\mathcal{D}_{wm}$. More formally, let $\mathcal{L}_{train}$ and $\mathcal{L}_{wm}$ be the loss functions corresponding to $\mathcal{D}_{train}$ and $\mathcal{D}_{wm}$, respectively. We start with initial parameters $\boldsymbol{\theta_0}$ and learning rate η. In each training epoch, we update the model parameters as follows:

1) Compute the gradient of $\mathcal{L}_{train}$ with respect to the model parameters ($\boldsymbol{\theta}$) using $\mathcal{D}_{train}$.
2) Update $\boldsymbol{\theta}$ by applying the gradients from step (1) using an optimizer.
3) Compute gradient of $\mathcal{L}_{wm}$ with respect to $\boldsymbol{\theta}$ using $\mathcal{D}_{wm}$.

4) Update $\boldsymbol{\theta}$ by applying the gradients from step (3) using the same optimizer.

Our approach of backpropagating losses from $\mathcal{D}_{train}$ and $\mathcal{D}_{wm}$ separately is motivated by the intuition that it allows the model to effectively learn both distributions independently. By first updating $\boldsymbol{\theta}$ based on $\mathcal{D}_{train}$ allows the model to capture the inherent patterns and relationships present in the non-watermarked instances. Then, updating $\boldsymbol{\theta}$ based on $\mathcal{D}_{wm}$ enables the model to associate the watermark patterns with the predetermined incorrect predictions. Consequently, it embeds the watermark information effectively.

4.3. Watermark verification

The watermark verification process requires testing $\mathcal{D}_{wm}$ against $\mathcal{M}_{adv}$. We expect GENIE to have the property of non-trivial ownership, i.e., if $\mathcal{M}_{adv}$ is $\mathcal{M}_{wm}$, then $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ should be high, otherwise $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ should be low. Therefore, the watermark verification process requires: (1) an assurance of GENIE satisfying the property of non-trivial ownership; and (2) a procedure to calculate the specific threshold as to how high or low should $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ be, to classify $\mathcal{M}_{adv}$ as $\mathcal{M}_{wm}$.

We first demonstrate the non-trivial ownership property of GENIE using statistical hypothesis testing in §4.3.1. We then outline problems with existing watermark threshold setting procedure as a motivation for DWT (our novel threshold setting procedure) and then describe its mechanism in §4.3.2.

4.3.1. Non-trivial ownership. Following the works [12, 26, 34], we employ a **statistical-cum-empirical** approach in lieu of a theoretical approach to give assurance of GENIE satisfying the watermarking requirements (cf. §3). To provide such a statistical assurance, we use the Smoothed Bootstrap Approach (SBA) [35]. The reason to choose this test instead of conventional hypothesis tests (i.e., the parametric Welch's t-test [36] or the non-parametric Mann-Whiteney U test [37]) is twofold. First, SBA generalizes to non-normal data. Thus, it can be used in situations where all AUC scores are not normally distributed. SBA is applicable to our data as **the Shapiro-Wilk Test [38] with a significance level of $\alpha = 0.05$ for some of our scores rejects the null hypothesis of the data being normally distributed**[4]. Second, the Mann-Whiteney U test (among other non-parametric hypothesis tests) is a test to establish stochastic inequality of the distribution of the given two samples. In our case, it is trivially seen as all $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ are found to be greater than $AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{wm}}$. It means that performing non-parametric tests would give a positive result supporting our claim, i.e., the trivial p-value of 0, in all cases.

[4]The authors in [12, 34] consider data to be normally distributed.

Let $\mathcal{W}_i$ and $\mathcal{C}_i$, $i \in [n]$ denote n watermarked and clean models, respectively. Let α_i and β_i, $i \in [n]$ denote the AUC scores of models $\mathcal{W}_i$ and $\mathcal{C}_i$ on $\mathcal{D}_{wm}$, respectively. Our goal is to provide a statistical assurance that the difference between $\overline{\alpha}$ and $\overline{\beta}$ is significant, where $\overline{\alpha}$ and $\overline{\beta}$ denote the mean of α_i and β_i, $i \in [n]$, respectively. To this end, our null hypothesis $\mathbf{H_0}$ and alternate hypothesis $\mathbf{H_a}$ are shown in Eq. 4 and Eq. 5, respectively:

$$\mathbf{H_0} : \overline{\alpha} - \overline{\beta} = 0. \tag{4}$$

$$\mathbf{H_a} : \overline{\alpha} - \overline{\beta} > 0. \tag{5}$$

The theoretical analysis above yields a condition under which $\mathcal{O}$ can reject $\mathbf{H_0}$ with τ confidence level (or, $1-\tau$ significance level). We verify our analysis by assessing SEAL architecture's performance on Yeast dataset [39]; TABLE 2 present the results for $n = 10$. Applying the Shapiro-Wilk test on these results, we get the p-values of β_i and α_i, $i \in [n]$ to be 0.001 and 0.339, respectively. With a significance level of 0.05, we reject the null hypothesis of β_i, $i \in [10]$ to be normally distributed. We perform smoothed bootstrap with the number of bootstrap samples equal to 10^5, bandwidth set according to Silverman's rule of thumb [40], and get the p-value to be 0.000. With the significance level of 0.05, we reject the hypothesis $\mathbf{H_0}$ for SEAL architecture on the Yeast dataset. The results for every model architecture and dataset considered in our work are presented in Appendix A; where for each architecture-dataset combination, we reject $\mathbf{H_0}$ with the significance level of 0.05 as well. Thus, **GENIE satisfies the non-trivial ownership requirement** (described in §3).

4.3.2. Dynamic Watermark Thresholding (DWT). Existing watermark threshold setting procedures can mainly be classified into 3 types [8]: (1) choosing the highest $AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{wm}}$ as the threshold; (2) choosing the lowest $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ as the threshold; (3) averaging $AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{wm}}$ and $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ of multiple $\mathcal{M}_{clean}$ and $\mathcal{M}_{wm}$ models. Procedures (1) and (2) will have high FPR and FNR, respectively. Procedure (3) tries to balance the trade-off of FPR and FNR present in procedures (1) and (2), but it does not give any statistical assurance. Regardless, all these procedures suffer from various other problems, like the lack of efficiency, requiring to train as many as 400 independent models (e.g., in work [7]), lack of theoretical or statistical assurance (e.g., in works [7, 8]), lack of generalizability to other watermarking schemes (e.g., in work [9]), and the naive assumption of normality of data (e.g., in works [10, 12, 34]). These

TABLE 2. AUCs OF THE WATERMARKED AND CLEAN MODELS ON THE YEAST DATASET WITH SEAL ARCHITECTURE ($n = 10$).

Score	AUC (%)									
	i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	i=9	i=10
β_i	14.37	6.73	12.49	15.54	10.21	8.03	4.23	40.05	5.02	10.72
α_i	97.50	98.02	98.09	97.75	97.83	97.21	97.47	97.15	97.87	97.96

problems greatly limit a watermarking scheme's practical applicability. Considering these issues, we identify the following four main properties of an ideal watermark threshold setting procedure:

1) **Efficiency**: The watermark threshold setting procedure should utilize as few $\mathcal{M}_{clean}$ and $\mathcal{M}_{wm}$ as possible.
2) **Assurance**: The procedure should be able to give assurance of its correctness, either theoretically or statistically.
3) **Generality**: The procedure should: (a) generalize to all kinds of watermarking scheme and model architectures; and (b) make as few assumptions about the data as possible, i.e., it should generalize to all distributions of data (viz., normal, non-normal).
4) **Robustness**: The watermark thresholds obtained from the procedure should be robust to outliers present in the data, i.e., in $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{clean}}$ and $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$.

To the best of our knowledge, none of the existing works have a watermark threshold setting procedure that complies to all of the aforementioned properties. We introduce DWT - a simple, yet effective watermark threshold setting procedure that encapsulates all the four identified properties.

The general idea of DWT is to use Kernel Density Estimate (KDE) to estimate the distribution of $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{clean}}$ and $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$. It draws a *random sample of size* n [41] from the two KDE distributions of $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{clean}}$ and $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$, and picks a threshold t that minimizes FPR and FNR. If a t is found such that the FPR and FNR is 0 across multiple random sample draws, a statistical assurance can be given that FPR and FNR are both less than n^{-1}. By using KDE, we make minimal assumptions regarding the distribution of data. Therefore, DWT would work irrespective of whether the underlying data distribution is normal or not. To summarize, DWT encapsulates the following characteristics: (1) it requires as few as 10 independent $\mathcal{M}_{clean}$ and $\mathcal{M}_{wm}$ to set a threshold; (2) it gives statistical assurance of FPR and FNR being less than n^{-1}; (3) it generalizes to all data distributions and watermarking scheme using accuracy or $AUC_{\mathcal{D}}^{\mathcal{M}}$ as the basis for deciding ownership; and (4) it is robust to outliers encountered in the data, i.e., the threshold changes *dynamically* in response to outliers present in data.

In a practical setting, t only needs to be calculated by $\mathcal{J}$ in case of a dispute (cf. §4.4). Considering most models will not encounter ownership disputes, we consider the expected computational cost involved in the calculation of t to be reasonable. Figure 4 illustrates DWT using Yeast with SEAL for setting threshold. The threshold for each dataset-architecture pair using DWT is listed in TABLE 3. Please note that a watermark threshold for a dataset-architecture pair appearing "low" (e.g., arXiv-GCN, NS-SEAL) may raise concern of OD falsely classifying $\mathcal{M}_{clean}$ as $\mathcal{M}_{wm}$, i.e., higher FPR. Despite the concern, the watermark thresholds can still be used for OD as DWT ensures FPR and FNR to be low. The verity of these "low" watermark thresholds can also be empirically justified from the corresponding $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{clean}}$ being "low" in TABLEs A.1, A.2, A.3, and A.4 in Appendix A.

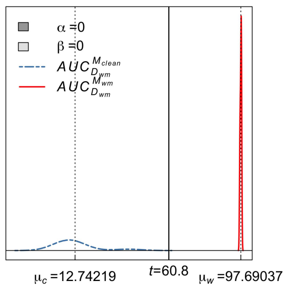


Figure 4. KDEs of $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{clean}}$ and $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ for dataset-architecture pair Yeast-SEAL, along with the corresponding watermark threshold t with $n = 10^6$. α and β here denote the FPR and FNR (not visible since they are 0), while μ_c and μ_w denote the *population* mean of $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{clean}}$ and $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$.

TABLE 3. WATERMARK THRESHOLD FOR GENIE ACROSS DIFFERENT MODELS AND DATASETS.

Dataset	AUC (%)			
	SEAL	GCN	GraphSAGE	NeoGNN
C.ele [42]	48.90	50.65	39.35	38.42
USAir [43]	10.56	49.69	40.07	18.02
NS [44]	5.06	64.82	41.69	41.44
Yeast [39]	60.80	42.35	66.45	12.63
Power [42]	40.55	52.29	53.04	54.85
arXiv [45]	12.27	10.00	28.96	16.22
PPI [46]	35.80	32.77	40.74	36.76

4.4. Ownership Demonstration (OD)

We now outline the process for $\mathcal{O}$ to demonstrate her ownership over $\mathcal{A}$'s model (i.e., $\mathcal{M}_{adv}$). OD uses $\mathcal{J}$ briefly outlined in §3. GENIE has a two-step OD procedure that involves a *model registration* step and a *dispute resolution* step.

Model registration: As a preemptive step of OD, $\mathcal{O}$ first sends $\mathcal{G}$ to $\mathcal{J}$ to procure $\mathcal{D}_{wm}$, addressing the problem of malicious plaintiff outlined in [8]. $\mathcal{J}$ then writes $\mathcal{D}_{wm}$ onto a time-stamped public bulletin board (e.g., blockchain) to provide the proof of anteriority in case of a dispute. We call this preemptive step model registration since it is analogous to patent registration common in the protection of IP rights [47, 48]. The procured $\mathcal{D}_{wm}$ will then be used for embedding the watermark into $\mathcal{M}$, i.e., train an untrained $\mathcal{M}$ to be $\mathcal{M}_{wm}$ by $\mathcal{O}$.

Dispute resolution: When a dispute arises, OD involves the following steps: (1) $\mathcal{O}$ accuses $\mathcal{A}$ of plagiarising her model $\mathcal{M}_{wm}$; (2) $\mathcal{A}$ sends $\mathcal{M}_{adv}$ to $\mathcal{J}$ for an evaluation; (3) $\mathcal{O}$ sends $\mathcal{D}_{wm}$ and the hashes of all the files. Here, the hashes are sent via a secure communication channel to ensure that the files are not tampered with; (4) $\mathcal{J}$ runs a check on the hashes of the files sent. Next, $\mathcal{J}$ checks the record of $\mathcal{D}_{wm}$ in the public bulletin board. If a matching record is found, $\mathcal{J}$ first calculates the watermark threshold t, and evaluates $\mathcal{M}_{adv}$ on $\mathcal{D}_{wm}$ to get $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$. The OD ends with a comparison of $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ against t, settling the dispute between $\mathcal{O}$ and $\mathcal{A}$ with a just verdict. On the other hand, if a record is not found, the dispute resolves in $\mathcal{O}$'s defeat.

5. Evaluation

We evaluate GENIE using 7 real-world datasets on 3 model architectures. We describe our experimental setup in §5.1. We evaluate the functionality preservation requirement of GENIE in §5.2 and compare our watermark embedding method with previous methods in §5.3. The robustness of GENIE is tested against 3 model extraction attacks (i.e., soft, hard, and double extraction; cf. §5.4.1) and 11 state-of-the-art backdoor defenses (i.e., 1 knowledge distillation (cf. §5.4.2), 4 model fine-tuning methods (cf. §5.4.3), 2 model compression techniques (cf. §5.4.4), 4 fine-pruning (cf. §5.4.5)). We further evaluate non-ownership piracy and efficiency requirements of GENIE in §5.5 and §5.7, respectively. Finally, we discuss a defense strategy to counter adaptive attackers in §5.6.

5.1. Experimental setup

We run all our experiments on an NVIDIA DGX A100 machine using Pytorch [49] framework. In what follows, we describe the datasets, models, and metrics used in our work.

Datasets: Following prior works [18, 50], we use 7 publicly available real-world graph datasets of varying sizes and sparsities in our experiments: USAir, NS, Yeast, C.ele, Power, arXiv and PPI (cf. Appendix B.1 for dataset details).

It is worth highlighting that none of these datasets, except PPI, have node attributes. Since GENIE uses node attributes for watermark embedding, we generate node features using Node2Vec [50] for all the datasets. We follow an 80-10-10 train-validation-test split of all the datasets across all our experiments. We use *Adam* optimizer and *negative log likelihood* loss for model training. Please refer Appendix B for our watermarking rates.

Models: We implement GENIE for SEAL [18] (SotA subgraph-based LP method) and for NeoGNN [51] (SotA node-representation based LP method). We also implement GENIE for GCN [19] and GraphSAGE [52] that are widely used GNN architectures (cf. Appendix B.2 for further details).

Metric: We use AUC across all our experiments to evaluate the performance of GENIE. AUC is threshold independent and is a widely used metric for binary classification tasks, such as LP. AUC can be interpreted as the probability of a given classifier ranking a randomly chosen instance of a positive class higher than a randomly chosen instance of a negative class. Accordingly, a random classifier will have an AUC score of 0.5 [53].

5.2. Functionality preserving

To evaluate GENIE's functionality-preserving nature, we compare the performance of $\mathcal{M}_{clean}$ and $\mathcal{M}_{wm}$ on $\mathcal{D}_{test}$. We assess GENIE's effectiveness by evaluating $\mathcal{M}_{wm}$'s performance on $\mathcal{D}_{wm}$. We establish a strict threshold of 2% drop from $AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{test}}$ to $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$ as the criterion for a watermarking scheme to be considered functionality-preserving. A high AUC score on $\mathcal{D}_{wm}$ indicates that the model has successfully learned to associate watermark patterns with predetermined incorrect predictions, enabling reliable watermark detection and ownership verification. We set 80% as the minimum $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ for reliable watermark detection. TABLE 4 presents $AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{test}}$, $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$, and $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ across all 7 datasets and 4 model architectures considered in our work. The reported scores are averaged from 10 runs using different random seeds to mitigate potential seed influence and capture GENIE's real mean performance. Our results show less than a 2% decrease in AUC scores on $\mathcal{D}_{test}$ due to watermarking across all model architectures and datasets, allowing us to claim that **GENIE is functionality preserving**. Furthermore, $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ exceeds 80% across all model architectures and datasets, indicating that $\mathcal{M}_{wm}$ has successfully learned $\mathcal{F}_{wm}$. Interestingly, we observe an increase in AUC scores on $\mathcal{D}_{test}$ after watermarking in some cases. This improvement could result from the watermark embedding process acting as a regularizer, potentially reducing the model's overfitting tendency.

5.3. Comparison among embedding methods

We compare our watermark embedding method with 4 baselines: (1) **Fine-Tuning**: Xu et al. [12] fine-tunes a clean pre-trained model on $\mathcal{D}_{wm}$. We fine-tune for 50 epochs to avoid over-fitting on $\mathcal{D}_{wm}$. (2) **Data Poisoning**: Adi et al. [5] adds trigger samples to the training set to embed the watermark. (3) **Uniform Loss**: Minimizing the sum of loss functions ($\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$). (4) **MGDA [54]**: It computes scaling coefficients α_1, α_2 for losses $\mathcal{L}_1, \mathcal{L}_2$, respectively by minimizing the weighted sum of losses ($\mathcal{L} = \alpha_1\mathcal{L}_1 + \alpha_2\mathcal{L}_2 \mid \alpha_1 + \alpha_2 = 1$).

We define that a practical watermark embedding method ensures a less than 2% drop from $AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{test}}$ to $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$ and achieves a minimum of 80% $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ (cf. §5.2). Our empirical results in TABLE 5 reveal that the method by

TABLE 4. WATERMARK VERIFICATION PERFORMANCE (AVERAGE OF 10 RUNS) OF GENIE ACROSS 4 MODEL ARCHITECTURES AND 7 DATASETS.

Dataset	SEAL			GCN			GraphSAGE			NeoGNN		
	$AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{test}}$ (%)	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$ (%)	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ (%)	$AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{test}}$ (%)	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$ (%)	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ (%)	$AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{test}}$ (%)	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$ (%)	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ (%)	$AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{test}}$ (%)	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$ (%)	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ (%)
C.ele	87.84 ± 0.46	87.60 ± 0.10	84.28 ± 0.93	88.97 ± 0.44	87.93 ± 0.43	100 ± 0.00	86.76 ± 0.68	85.71 ± 0.87	100 ± 0.00	89.03 ± 0.71	88.94 ± 1.20	100 ± 0.00
USAir	93.19 ± 0.25	93.64 ± 0.17	92.29 ± 0.58	90.02 ± 0.52	89.35 ± 0.72	100 ± 0.00	92.44 ± 0.35	92.29 ± 0.65	100 ± 0.00	95.81 ± 0.81	94.57 ± 1.45	100 ± 0.00
NS	98.10 ± 0.15	98.11 ± 0.23	98.70 ± 0.03	95.44 ± 0.74	96.26 ± 0.88	99.78 ± 0.00	90.90 ± 0.63	93.66 ± 0.47	99.78 ± 0.00	99.93 ± 0.02	99.80 ± 0.14	100 ± 0.00
Yeast	97.07 ± 0.21	97.38 ± 0.16	97.69 ± 0.33	93.64 ± 0.40	91.73 ± 0.39	100 ± 0.00	89.12 ± 0.43	90.70 ± 0.43	100 ± 0.00	97.78 ± 0.57	97.54 ± 0.19	100 ± 0.00
Power	84.41 ± 0.44	83.91 ± 0.25	88.28 ± 0.03	99.36 ± 0.17	99.12 ± 0.19	99.00 ± 0.00	87.54 ± 1.02	92.68 ± 1.06	99.00 ± 0.00	99.96 ± 0.02	99.94 ± 0.04	100 ± 0.00
arXiv	98.14 ± 0.14	97.17 ± 0.49	98.15 ± 0.16	99.31 ± 0.04	98.78 ± 0.15	99.99 ± 0.00	99.62 ± 0.01	99.32 ± 0.13	99.99 ± 0.00	99.92 ± 0.01	99.91 ± 0.01	94.22 ± 3.99
PPI	89.63 ± 0.12	89.45 ± 0.16	84.28 ± 1.38	95.08 ± 0.04	94.82 ± 0.05	100 ± 0.00	94.03 ± 0.09	94.31 ± 0.16	100 ± 0.00	97.43 ± 0.16	97.44 ± 0.11	97.64 ± 1.77

Xu et al. [12] violates the functionality-preserving property across all datasets, with a drop of more than 2% in model utility. While the method by Adi et al. [5] maintains functionality preservation, it performs poorly on $\mathcal{D}_{wm}$ for 4 out of 7 datasets. The uniform loss approach violates the functionality-preserving property for 3 out of 7 datasets. Despite MGDA [54] performing on par with or better than GENIE in some cases, it fails to preserve functionality for large datasets (i.e., arxiv, PPI). This comparative analysis demonstrates the superiority of GENIE's watermark embedding method over existing approaches. GENIE consistently preserves functionality while achieving high performance on $\mathcal{D}_{wm}$ across all datasets.

TABLE 5. COMPARISON AMONG WATERMARK EMBEDDING METHODS.

Dataset		C.ele	USAir	NS	Yeast	Power	arXiv	PPI
No Watermark	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{clean}}$ (%)	87.90	89.62	96.00	93.45	99.54	99.28	95.83
Xu et al. [12]	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	53.41	33.32	78.57	49.80	74.74	20.88	26.28
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)	100	99.32	95.56	100	91.50	100	100
Adi et al. [5]	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	**88.50**	**91.45**	**96.23**	**94.06**	**99.28**	**99.06**	**95.95**
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)	90.62	74.51	96.67	53.25	98.00	4.38	21.58
Uniform Loss ($\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2$)	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	88.37	80.53	96.44	92.52	**99.20**	93.29	91.45
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)	100	98.83	99.78	100	99.00	100	100
MGDA [54]	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	**88.57**	**89.93**	95.76	**92.82**	99.04	96.40	93.81
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)	100	100	99.78	100	99.00	100	100
GENIE's approach	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	86.93	88.34	**96.59**	91.46	98.92	**98.13**	**94.67**
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)	100	100	99.77	100	99.00	100	100

Note: The highest and the second highest $AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ values are bold and underlined, respectively. This table shows our ablation done using GCN model.

5.4. Robustness

To ensure the robustness and reliability of a watermarking scheme, it is crucial to assess its resilience against potential attempts by $\mathcal{A}$ to watermark. Since $\mathcal{A}$ cannot directly verify the presence of watermark in $\mathcal{M}_{wm}$, $\mathcal{A}$ may resort to various techniques (e.g., model extraction, model pruning, fine-tuning) to eliminate or degrade the embedded watermark. We show the robustness of GENIE against various watermark-removal techniques in the following sections. $\mathcal{A}$ might fail or succeed while trying to remove the watermark from $\mathcal{M}_{wm}$. We define the success and failure of GENIE as:

Success: If $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ is above the watermark threshold after a watermark removal attempt, we identify it as a watermark success since we can verify the model's ownership. If $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ drops below the watermark threshold and $AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ drops by more than **10%**, we still consider it as a watermark success since $\mathcal{A}$ is losing the model's utility in exchange of watermark removal attempt.

Failure: If $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ drops below the watermark threshold and $AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ doesn't drop more than 10%, we consider it as a watermark failure since $\mathcal{A}$ was successful in removing the watermark without much loss of the model's utility.

Due to the page length limit, here we present the results from robustness tests on GENIE for GCN only. We detail the corresponding results for other models in Appendix C.

5.4.1. Model extraction attacks.

Such attacks [1–3] pose a significant threat to DNNs as they enable an adversary to steal the functionality of a victim model. In these attacks, $\mathcal{A}$ queries the victim model (i.e., $\mathcal{M}_{wm}$ in our case) using publicly available test samples and collect responses to train a surrogate model (i.e., $\mathcal{M}_{adv}$) to steal $\mathcal{M}_{wm}$'s functionality. The literature on model extraction attacks is limited in the context of LP tasks on GNNs. Therefore, to evaluate GENIE against model extraction attacks, we modify the loss function employed in the knowledge distillation process [55] as outlined in Eq. 6 and Eq. 7.

$$\mathcal{L}_{soft} = \mathcal{L}_{CE}\left(\phi\left(\theta_{wm}\right), \phi\left(\theta_{adv}\right)\right). \tag{6}$$

$$\mathcal{L}_{hard} = \mathcal{L}_{CE}\left(\hat{y}\left(\theta_{wm}\right), \hat{y}\left(\theta_{adv}\right)\right). \tag{7}$$

Here, θ_{wm} and θ_{adv} denote the model parameters of $\mathcal{M}_{wm}$ and $\mathcal{M}_{adv}$, $\phi(\theta_{wm})$ and $\phi(\theta_{adv})$ represent the logits (i.e., output scores) produced by $\mathcal{M}_{wm}$ and $\mathcal{M}_{adv}$, while $\hat{y}(\theta_{wm})$ and $\hat{y}(\theta_{adv})$ denote the hard predictions (e.g., 0 or 1) made by the respective model. $\mathcal{L}_{CE}$ denotes cross-entropy loss.

We consider 3 types of model extraction techniques, viz., soft label, hard label, and double extraction. In soft label extraction, we apply $\mathcal{L}_{CE}$ between the logits of $\mathcal{M}_{wm}$ and $\mathcal{M}_{adv}$ to train $\mathcal{M}_{adv}$ (cf. Eq. 6). In hard label extraction, we apply $\mathcal{L}_{CE}$ between the predictions of $\mathcal{M}_{wm}$ and $\mathcal{M}_{adv}$ to train $\mathcal{M}_{adv}$ (cf. Eq. 7). In double extraction, we perform the hard label extraction twice to obtain the final $\mathcal{M}_{adv}$. Double extraction is a tougher setting since it is difficult for the watermark to survive model extraction twice. We train $\mathcal{M}_{adv}$ model using half of $\mathcal{D}_{test}$ and evaluate it with the other half. Table 6 shows $AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$, $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ before model extraction and $AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$, $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ after model extraction attack.

TABLE 6. IMPACT OF MODEL EXTRACTION.

Dataset		C.ele	USAir	NS	Yeast	Power	arXiv	PPI
Before model extraction	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	86.93	88.34	96.59	91.46	98.92	98.13	94.67
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)	100	100	99.77	100	99.00	100	100
After soft extraction	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	87.35	89.00	95.74	91.57	98.21	98.28	94.93
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	90.62	96.87	95.77	100	94.00	90.18	97.33
After hard extraction	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	88.27	88.43	95.95	90.76	96.80	97.74	93.91
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	82.81	90.91	94.88	100	90.99	82.35	87.14
After double extraction	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	84.08	86.91	66.99	89.07	80.51	96.99	93.34
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	65.62	78.32	65.55	92.30	77.00	43.43	53.85

We observe a maximum drop of 18.19% from $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ to $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (for the C.ele dataset) under hard extraction attack, and a maximum drop of 9.82% (for the arXiv dataset) under soft extraction attack. Despite these drops, $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ remains significantly above the watermark threshold (cf. TABLE 3) in both cases, which ensures reliable ownership verification. We note a greater drop from $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ to $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ under hard extraction attack compared to soft extraction attack, which is naturally expected since logits provide richer information about the decision boundary than hard predictions. Despite a more significant drop from $AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ to $AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ and from $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ to $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ under double extraction attack, $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ still remains substantially above the watermark threshold (cf. TABLE 3) across all datasets. It demonstrates GENIE's robustness against persistent model extraction attempts.

Different model architectures: It is also possible that $\mathcal{A}$ might not choose the same architecture to steal the model via model extraction. TABLE 7 presents the outcomes of all 3 model extraction attacks when the $\mathcal{M}_{adv}$'s architecture (i.e., GraphSAGE) differs from $\mathcal{M}_{wm}$'s architecture (i.e., GCN). We find that $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ is still above the watermark threshold (cf. TABLE 3) under all attacks across all datasets; except for the power dataset under double extraction attack. However, $AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ in that case drops from 98.93% to 58.81%, rendering the attack useless. Thus, we can conclude that **GENIE is robust against model extraction attacks even when $\mathcal{M}_{adv}$ and $\mathcal{M}_{wm}$ have different architecture**.

TABLE 7. IMPACT OF MODEL EXTRACTION WHEN ARCHITECTURE OF $\mathcal{M}_{adv}$ (I.E., GRAPHSAGE) DIFFERS FROM $\mathcal{M}_{wm}$ (I.E., GCN).

Dataset		C.ele	USAir	NS	Yeast	Power	arXiv	PPI
Before model extraction	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	86.93	88.34	96.59	91.46	98.92	98.13	94.67
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)	100	100	99.77	100	99.00	100	100
After soft extraction	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	87.14	88.89	78.52	90.42	70.51	98.29	94.61
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	96.88	97.56	94.89	99.41	91.00	83.05	92.10
After hard extraction	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	89.64	88.05	75.91	88.75	70.63	97.59	93.71
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	93.75	95.31	98.00	89.94	91.00	71.93	83.65
After double extraction	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	85.38	85.07	59.86	85.59	**58.81**	96.23	91.84
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	79.69	78.71	66.67	67.46	49.50	41.90	47.75

5.4.2. Knowledge distillation. It is the process of transferring knowledge from a teacher model to a student model [55]. In our context, the teacher is $\mathcal{M}_{wm}$ and the student is $\mathcal{M}_{adv}$. The extraction process comprises training $\mathcal{M}_{adv}$ on the logits of $\mathcal{M}_{wm}$ and the ground truth [55]. It helps with decreasing the overfitting of the victim model (i.e., $\mathcal{M}_{wm}$). Consequently, $\mathcal{A}$ might be able to remove the watermark and reproduce the core model functionality.

To test GENIE's robustness, we assume that the student model have the same architecture as the teacher model. Moreover, we use half of $\mathcal{D}_{test}$ for distillation and evaluate $\mathcal{M}_{adv}$ on the other half. TABLE 8 presents $AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$, $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ before knowledge distillation and $AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$, $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ after knowledge distillation.

TABLE 8. IMPACT OF KNOWLEDGE DISTILLATION.

Dataset		C.ele	USAir	NS	Yeast	Power	arXiv	PPI
Before distillation	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	86.93	88.34	96.59	91.46	98.92	98.13	94.67
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)	100	100	99.77	100	99.00	100	100
After distillation	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	88.60	89.39	95.00	92.19	98.54	98.71	95.25
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	81.25	86.33	90.89	98.22	86.00	74.76	94.95

We observe a maximum drop of 25.24% from $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ to $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (for the arXiv dataset). It important to note that $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ still remains significantly above the watermark threshold (cf. TABLE 3), which indicates that $\mathcal{A}$ was not successful in removing the watermark using knowledge distillation. To summarize, our results show that knowledge distillation was able to transfer the core functionality of the victim model, but the watermark was transferred too (as $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ is still above the threshold for all datasets). We can conclude that **GENIE is robust against knowledge distillation.**

5.4.3. Model fine-tuning. Fine-tuning [56] is one of the most commonly used attacks to remove the watermark since it is computationally inexpensive and does not compromise the model's core functionality much. To test GENIE against this attack, we use half of $\mathcal{D}_{test}$ for fine-tuning and evaluate the fine-tuned model's (i.e., $\mathcal{M}_{adv}$) performance with the other half. Through extensive experimentation and analysis, we determined that limiting the training process to 50 epochs serves as an optimal strategy (i.e., to avoid the risk of overfitting $\mathcal{M}_{adv}$ on the subset of $\mathcal{D}_{test}$ used for fine-tuning). We evaluate GENIE against 4 variations of fine-tuning. These can be classified into two broad categories [56]:

Last layer fine-tuning: This fine-tuning procedure updates the weights of only the last layer of the target model. It can be done in the following two ways.

1) Fine-Tune Last Layer (**FTLL**): Freezing the weights of the target model, updating the weights of its last layer only during fine-tuning.
2) Re-Train Last Layer (**RTLL**): Freezing the weights of the target model, reinitializing the weights of only its last layer, and then fine-tuning it.

All layers fine-tuning: This fine-tuning procedure updates weights of all the layers of the target model. It is a stronger setting compared to the last layer fine-tuning method as all the weights are updated, which makes it tougher to retain the watermark. It can be done in the following two ways.

1) Fine-Tune All Layers (**FTAL**): Updating weights of all the layers of the target model during fine-tuning.
2) Re-Train All Layers (**RTAL**): Reinitializing the weights of target model's last layer, updating weights of all its layers during fine-tuning.

FTLL is considered the weakest attack because it has the least capacity to modify the core GNN layers responsible for learning the watermark. Conversely, RTAL is considered the toughest attack because it enables complete fine-tuning of all model layers, providing the highest flexibility to potentially overwrite or distort the watermark embedded across multiple layers. TABLE 9 lists $AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$, $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ before fine-tuning (in column with $\star$) and $AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$, $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ after fine-tuning with all the four types.

TABLE 9. IMPACT OF MODEL FINE-TUNING.

Dataset		Fine-tuning method				
		No fine-tuning*	FTLL	RTLL	FTAL	RTAL
C.ele	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	86.93	82.46	70.61	74.79	68.51
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	90.62	60.94	73.44	53.12
USAir	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	88.34	89.09	87.68	86.79	84.68
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	90.62	81.64	80.52	69.24
NS	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	96.59	98.70	98.59	89.99	73.95
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	99.77	99.78	96.22	93.56	71.78
Yeast	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	91.46	91.61	90.79	90.33	87.08
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	91.72	89.35	98.22	63.91
Power	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	98.92	99.39	99.26	97.74	95.05
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	99.00	99.00	99.00	77.00	73.00
arXiv	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	98.13	98.57	97.65	98.78	98.04
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	88.23	46.54	55.54	19.92
PPI	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	94.67	94.94	94.35	94.94	94.35
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	94.95	55.46	77.96	48.67

Column with * shows the values when $\mathcal{M}_{adv} = \mathcal{M}_{wm}$.

We note that our watermark survives against all fine-tuning procedures for all the datasets; since $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$

remains above the watermark threshold (cf. TABLE 3) in each case. Therefore, we conclude that **GENIE is robust against model fine-tuning**.

5.4.4. Model compression. It is a technique to reduce the size and complexity of a DNN, thereby making it more efficient and easily deployable. Compressing the model can inadvertently or otherwise act as an attack against the watermark. Thus, we test GENIE's robustness with following two model compression techniques:

Model pruning: Model or parameter pruning [57] selects a fraction of weights that have the smallest absolute value and makes them zero. It is a computationally inexpensive watermark removal technique. We evaluate GENIE against different pruning fractions starting from 0.2 at a step size of 0.2. TABLE 10 lists $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$, $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ before model pruning (in column with $\star$) and $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$, $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ after model pruning with different prune percentage. We see that even after pruning **80%**[5] of $\mathcal{M}_{wm}$'s weights, we are still able to verify the ownership from the resultant $\mathcal{M}_{adv}$ in almost every case. Given that $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ remains above the watermark threshold (cf. TABLE 3) roughly in all cases, we conclude that **GENIE is robust against model pruning**.

TABLE 10. IMPACT OF MODEL PRUNING.

Dataset		Prune Percentage (%)					
		No Pruning*	20	40	60	80	100
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	86.93	86.97	86.31	83.92	75.43	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100	100	100	100	70.31	50.00
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	88.34	88.98	89.57	89.21	78.50	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100	100	100	92.77	71.28	50.00
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	96.59	96.25	96.01	96.08	91.31	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.77	99.77	99.77	96.66	85.55	50.00
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.46	91.27	89.97	85.71	80.50	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100	100	100	100	74.55	50.00
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	98.92	99.00	99.20	98.32	92.43	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.00	99.00	95.00	74.00	55.00	50.00
arXiv	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	98.13	98.08	97.911	94.79	84.37	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100	100	99.98	83.09	41.95	50.00
PPI	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.67	94.60	94.05	92.86	90.06	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100	100	100	78.87	31.22	50.00

Column with * shows the values when $\mathcal{M}_{adv} = \mathcal{M}_{wm}$.

Weight quantization: It is another model compression technique to reduce the size of a model. It changes the representation of weights to a lower-bit system, thereby saving memory. It is often used to compress large models, e.g., LLMs [58]. We follow the standard weight quantization method [59] with bit-size = 3. TABLE 11 reports that $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ remains remarkably above the watermark threshold (cf. TABLE 3) after quantization in all the cases, making **GENIE is robust against weight quantization**.

TABLE 11. IMPACT OF WEIGHT QUANTIZATION.

Dataset		C.ele	USAir	NS	Yeast	Power	arXiv	PPI
Before quantization	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$ (%)	86.93	88.34	96.59	91.46	98.92	98.13	94.67
	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ (%)	100	100	99.77	100	99.00	100	100
After quantization	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	84.59	85.16	97.79	85.84	98.39	96.07	87.56
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	96.88	90.92	99.78	82.25	98.00	81.58	67.95

[5]A model obtained after 100% pruning is equal to a random classifier.

5.4.5. Fine-pruning. It is a key defense against a backdoor attack that combines model pruning and fine-tuning. It is more effective than individual pruning or fine-tuning, which makes it difficult for the watermark to survive. We start by pruning a fraction of the smallest absolute weights. Next, we fine-tune the pruned model with half of $\mathcal{D}_{test}$ and evaluate the pruned+fine-tuned model (i.e., $\mathcal{M}_{adv}$) with the other half. We perform an exhaustive evaluation with pruning fractions ranging from 0.2-0.8 at a step size of 0.2, which is followed by one of the four types of model fine-tuning (i.e., FTLL, RTLL, FTAL, RTAL). Our rigorous experiments aim to provide a holistic understanding of GENIE's robustness against the fine-pruning technique. TABLE 12 exhibits $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$, $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ before fine-pruning (in column with $\star$) and $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$, $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ after fine-pruning with different pruning fractions followed by RTAL fine-tuning method (cf. TABLEs C.1-C.3 in Appendix C.2 for fine-pruning via FTLL, RTLL, and FTAL, respectively).

We observe the highest drop from $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ to $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ when fine-pruning is performed using RTAL (cf. TABLE 12), which is expected as RTAL represents the strongest attack that enables complete fine-tuning of all model layers. Nevertheless, $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ remains above the watermark threshold (cf. TABLE 3) in most cases. We strongly believe that **GENIE is robust against fine-pruning** given that it is failing at only **2 out of 112** cases.

TABLE 12. IMPACT OF PRUNING + RTAL.

Dataset		Prune Percentage (%)				
		No Pruning*	20	40	60	80
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	86.93	69.11	66.34	72.06	68.28
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100	68.75	68.75	70.31	75.00
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	88.34	85.46	84.94	83.26	84.50
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100	63.76	65.52	62.79	47.75
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	96.59	76.49	79.69	74.29	74.74
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.77	78.00	85.11	78.88	78.88
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.46	85.70	85.47	84.50	82.89
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100	84.02	92.89	89.94	63.31
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	98.92	92.87	92.33	91.92	90.02
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.00	63.00	64.99	63.00	49.99
arXiv	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	98.13	98.02	98.01	97.61	95.82
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100	20.67	19.46	17.99	13.93
PPI	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.67	92.92	92.79	92.45	91.88
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100	42.51	44.81	46.74	27.08

Column with * shows the values when $\mathcal{M}_{adv} = \mathcal{M}_{wm}$.

5.5. Non-ownership piracy

Ownership piracy attacks allow $\mathcal{A}$ to follow GENIE to insert her pirated watermark into a model stolen from $\mathcal{O}$. This creates an ambiguity for $\mathcal{J}$ to determine which party has originally watermarked the given model. However, for such an attack to succeed, $\mathcal{A}$ must present $\mathcal{M}_{adv}$ containing only her pirated watermark, free of the $\mathcal{O}$'s original watermark. It is because if $\mathcal{M}_{adv}$ contains both $\mathcal{O}$'s and $\mathcal{A}$'s watermark, $\mathcal{J}$ can easily resolve the conflict by comparing the timestamps of both watermarks (cf. §4.4). However, owing to the robustness of GENIE (as witnessed in § 5.4), $\mathcal{A}$ will not be able to remove $\mathcal{O}$'s original watermark from $\mathcal{M}_{wm}$. It implies that $\mathcal{A}$'s success hinges on the possibility that embedding her watermark could overwrite (i.e., remove) $\mathcal{O}$'s watermark. In our series of experiments (cf. Appendix C.1), we observe that $\mathcal{O}$'s watermark cannot be overwritten after embedding $\mathcal{A}$'s watermark, which shows GENIE's robustness against ownership piracy attacks.

5.6. Dealing with adaptive attackers

Our results demonstrate GENIE's robustness against classical watermark removal techniques, including model extraction (cf. §5.4.1-§5.4.2), model overwriting (cf. §5.4.3-§5.4.5), and piracy attack (cf. §5.5). While these tests provide valuable insights into GENIE's overall robustness, it is crucial to evaluate its performance against newer attacks; in particular an adaptive attack, where $\mathcal{A}$ would design and implement an attack specifically tailored to GENIE.

To simulate real-world scenario, we assume that $\mathcal{A}$ accesses $\mathcal{M}_{own}$ (i.e., $\mathcal{M}_{wm}$) through an MLaaS system, querying $\mathcal{M}_{own}$ using only $\mathcal{V}$. It is analogous to the standard assumption of the user of an MLaaS being oblivious to its underlying complexities (e.g., a user avails the service of a recommendation MLaaS system using only node IDs, i.e., $\mathcal{V}$, while being oblivious to the underlying $(\mathbf{A},\ \mathbf{X})$). We summarise the state of an adaptive $\mathcal{A}$ as follows: (1) $\mathcal{A}$ has access to $\mathcal{M}_{adv}$ (i.e., $\mathcal{M}_{wm}$) apart from $\mathcal{O}$'s MLaaS; (2) $\mathcal{A}$ understands GENIE and knows that $\mathcal{M}_{adv}$ has been watermarked using GENIE; (3) $\mathcal{A}$ knows the watermarking rate used (viz., α_{nr} or α_{sg}); and (4) $\mathcal{A}$ knows the original graph $\mathcal{G}$.

The only information which $\mathcal{A}$ cannot infer from knowing GENIE are $\mathcal{D}_{wm}$ and $\mathcal{G}_{wm}$, which are secret, since they are created by random sampling. We discuss the viability of an adaptive attack that exploits this information for node representation and subgraph-based methods of GENIE as follows:

Node representation-based methods: In hopes to break GENIE (i.e., to remove the watermark from $\mathcal{M}_{adv}$), $\mathcal{A}$ may attempt to guess links present in $\mathcal{D}_{wm}$ and construct $\mathcal{G}_{wm}$ by continuously querying $\mathcal{O}$'s MLaaS system. If successful, $\mathcal{A}$ can compare $\mathcal{G}_{wm} = (\mathcal{V},\ \mathcal{E}_{wm})$ with $\mathcal{G} = (\mathcal{V},\ \mathcal{E})$ to get the randomly sampled watermark links $\mathcal{S}_{wm} = (\mathcal{E} \backslash \mathcal{E}_{wm}) \cup (\mathcal{E}_{wm} \backslash \mathcal{E})$ and then fine-tune $\mathcal{M}_{adv}$ with the labels opposite to $\mathcal{S}_{wm}$, potentially removing the watermark.

To defend GENIE against such an attack, $\mathcal{O}$ can design the MLaaS system to invert the output whenever a user attempts to query the links in $\mathcal{S}_{wm}$. Consequently, any attempt to reconstruct $\mathcal{G}_{wm}$ by querying $\mathcal{O}$'s MLaaS system would only result in reconstruction of $\mathcal{G}$ instead of $\mathcal{G}_{wm}$. To conclude, such a defense closes all doors for $\mathcal{A}$ to guess $\mathcal{G}_{wm}$, thereby protecting $\mathcal{O}$ against such an adaptive $\mathcal{A}$.

Subgraph-based methods: If a link present in $\mathcal{D}_{wm}$ is queried to $\mathcal{O}$'s MLaaS system, the returned output will not be watermarked, i.e., the MLaaS system will classify the link correctly. It is because during inference: (1) $\mathcal{M}_{own}$ constructs the k-hop subgraph $\mathcal{G}_k$ surrounding the link; and (2) performs binary classification of $\mathcal{G}_k$. Since $\mathcal{G}_k$'s node feature vectors $\mathbf{x}_v$ have not been replaced with the secret watermark vector $\mathbf{w}$ present in $\mathcal{D}_{wm}$, $\mathcal{M}_{own}$ will output the correct prediction of the link. Therefore, the guessing of links present in $\mathcal{D}_{wm}$ by querying $\mathcal{O}$'s MLaaS system is infeasible. Consequently, the exploitation of $\mathcal{D}_{wm}$'s knowledge is not possible, in case of subgraph-based methods.

5.7. Efficiency

The computational efficiency of a watermarking scheme is crucial, as it directly impacts its cost-effectiveness and practical adoption. In GENIE, the computational overhead is primarily determined by the size of the trigger set, as each epoch requires separate backpropagation for both the training set and the trigger set. Table B.2 illustrates the computational overhead in terms of α_{sg} and α_{nr}, representing the relative sizes of the trigger sets for subgraph-based and node representation-based methods, respectively. Our analysis reveals that the maximum α_{sg} is 40% and the maximum α_{nr} is 15%, both occurring in smaller datasets. Notably, for large datasets, both α_{sg} and α_{nr} remain below 4%. It demonstrates GENIE's computational overhead is reasonable, especially for large-scale real-world applications.

6. Conclusion

Despite the tremendous success of GNNs in learning graph-structured data, protecting trained GNN models from model-stealing attacks is a critical issue. Existing GNN watermarking schemes focus either on node or graph classification tasks. In this paper, we propose GENIE, a backdoor-based watermarking scheme for GNNs tailored to LP task. We design GENIE for node representation-based and subgraph-based methods of LP. Our extensive evaluations show that GENIE is functionality-preserving. At the same time, GENIE is robust against SotA watermark removal techniques and model extraction attacks. The statistical assurance given by the OD procedure confirm a close to zero probability of misclassification in GENIE. We hope our work sets new research directions and benchmarks in the domain of model watermarking.

References

[1] Y. Shen, X. He, Y. Han, and Y. Zhang, "Model Stealing Attacks against Inductive Graph Neural Networks," in *IEEE S&P*, 2022, pp. 1175–1192.

[2] D. DeFazio and A. Ramesh, "Adversarial Model Extraction on Graph Neural Networks," *arXiv preprint:1912.07721*, 2019.

[3] B. Wu, X. Yang, S. Pan, and X. Yuan, "Model Extraction Attacks on Graph Neural Networks: Taxonomy And Realisation," in *ASIACCS*, 2022, pp. 337–350.

[4] Y. Li, H. Wang, and M. Barni, "A Survey of Deep Neural Network Watermarking Techniques," *Neuro-Computing*, vol. 461, pp. 171–193, 2021.

[5] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, "Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring," in *USENIX Security*, 2018, pp. 1615–1631.

[6] F. Boenisch, "A Systematic Review on Model Watermarking for Neural Networks," *Frontiers in Big Data*, vol. 4, p. 729663, 2021.

[7] P. Lv, H. Ma, K. Chen, J. Zhou, S. Zhang, R. Liang, S. Zhu, P. Li, and Y. Zhang, "Mea-Defender: A Robust Watermark against Model Extraction Attack," *arXiv preprint:2401.15239*, 2024.

[8] J. Liu, R. Zhang, S. Szyller, K. Ren, and N. Asokan, "False Claims Against Model Ownership Resolution," in *USENIX Security*, 2024, pp. 6885–6902.

[9] S. Szyller, B. G. Atli, S. Marchal, and N. Asokan, "Dawn: Dynamic Adversarial Watermarking of Neural Networks," in *ACM MM*, 2021, pp. 4417–4425.

[10] N. Lukas, E. Jiang, X. Li, and F. Kerschbaum, "SoK: How Robust is Image Classification Deep Neural Network Watermarking?" in *IEEE S&P*, 2022, pp. 787–804.

[11] X. Zhao, H. Wu, and X. Zhang, "Watermarking Graph Neural Networks by Random Graphs," in *IEEE ISDFS*, 2021, pp. 1–6.

[12] J. Xu, S. Koffas, O. Ersoy, and S. Picek, "Watermarking Graph Neural Networks Based on Backdoor Attacks," in *IEEE EuroS&P*, 2023, pp. 1179–1197.

[13] H. Zheng, H. Xiong, H. Ma, G. Huang, and J. Chen, "Link-Backdoor: Backdoor Attack on Link Prediction via Node Injection," *IEEE TCSS*, vol. 11, pp. 1816–1831, 2023.

[14] J. Dai and H. Sun, "A Backdoor Attack against Link Prediction Tasks with Graph Neural Networks," *arXiv preprint:2401.02663*, 2024.

[15] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee, "Billion-Scale Commodity Embedding for E-Commerce Recommendation in Alibaba," in *KDD*, 2018, pp. 839–848.

[16] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph Convolutional Neural Networks for Web-Scale Recommender Systems," in *KDD*, 2018, pp. 974–983.

[17] A. Lerer, L. Wu, J. Shen, T. Lacroix, L. Wehrstedt, A. Bose, and A. Peysakhovich, "Pytorch-Biggraph: A Large Scale Graph Embedding System," *MLSys*, pp. 120–131, 2019.

[18] M. Zhang and Y. Chen, "Link Prediction Based on Graph Neural Networks," *NeurIPS*, vol. 31, pp. 1–11, 2018.

[19] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv preprint:1609.02907*, 2016.

[20] J. Li, H. Shomer, H. Mao, S. Zeng, Y. Ma, N. Shah, J. Tang, and D. Yin, "Evaluating Graph Neural Networks for Link Prediction: Current Pitfalls and New Benchmarking," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[21] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor Learning: A Survey," *IEEE TNNLS*, vol. 35, pp. 5–22, 2022.

[22] W. Guo, B. Tondi, and M. Barni, "An Overview of Backdoor Attacks against Deep Neural Networks and Possible Defences," *IEEE Open Journal of Signal Processing*, vol. 3, pp. 261–287, 2022.

[23] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph Backdoor," in *USENIX Security*, 2021, pp. 1523–1540.

[24] J. Xu, M. Xue, and S. Picek, "Explainability-Based Backdoor Attacks against Graph Neural Networks," in *ACM WiseML*, 2021, pp. 31–36.

[25] 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 *ACM ASIACCS*, 2018, pp. 159–172.

[26] B. Kim, S. Lee, S. Lee, S. Son, and S. J. Hwang, "Margin-Based Neural Network Watermarking," in *ICML*, 2023, pp. 16 696–16 711.

[27] N. N. Daud, S. H. Ab Hamid, M. Saadoon, F. Sahran, and N. B. Anuar, "Applications of Link Prediction in Social Networks: A Review," *Elsevier JNCA*, vol. 166, p. 102716, 2020.

[28] L. A. Adamic and E. Adar, "Friends and Neighbors on the Web," *Social Networks*, vol. 25, no. 3, pp. 211–230, 2003.

[29] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, pp. 30–37, 2009.

[30] T. Oyetunde, M. Zhang, Y. Chen, Y. Tang, and C. Lo, "Boostgapfill: Improving the Fidelity of Metabolic Network Reconstructions through Integrated Constraint and Pattern-Based Methods," *Bioinformatics*, vol. 33, no. 4, pp. 608–611, 2017.

[31] F. A. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information Hiding - A Survey," *Proceedings of the IEEE*, vol. 87, pp. 1062–1078, 1999.

[32] J.-E. Ekberg, K. Kostiainen, and N. Asokan, "The Untapped Potential of Trusted Execution Environments on Mobile Devices," *IEEE S&P Magazine*, vol. 12, no. 4, pp. 29–37, 2014.

[33] S. Gallagher. (2023) Trusted Compute Base. [Online]. Available: https://learn.microsoft.com/en-us/azure/confidential-computing/trusted-compute-base?source=recommendations

[34] J. Tan, N. Zhong, Z. Qian, X. Zhang, and S. Li, "Deep Neural Network Watermarking against Model Extraction Attack," in *ACM MM*, 2023, pp. 1588–1597.

[35] B. Efron, "Bootstrap Methods: Another Look at the Jackknife," *The Annals of Statistics*, pp. 1–26, 1979.

[36] B. L. Welch, "The generalization of 'STUDENT'S'problem when several different population varlances are involved," *Biometrika*, vol. 34, no. 1, pp. 28–35, 1947.

[37] "Mann Whitney Test," in *The Concise Encyclopedia of Statistics*, 2008, pp. 327–329.

[38] S. S. Shapiro and M. B. Wilk, "An Analysis of Variance Test for Normality (Complete Samples)," *Biometrika*, vol. 52, pp. 591–611, 1965. [Online]. Available: http://www.jstor.org/stable/2333709

[39] C. Von Mering, R. Krause, B. Snel, M. Cornell, S. G. Oliver, S. Fields, and P. Bork, “Comparative Assessment of Large-Scale Data Sets of Protein–Protein Interactions,” *Nature*, vol. 417, pp. 399–403, 2002.

[40] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Routledge, 2018.

[41] H. Pishro-Nik, *Introduction to Probability, Statistics, and Random Processes*. Kappa Research, LLC Blue Bell, PA, USA, 2014.

[42] D. J. Watts and S. H. Strogatz, “Collective Dynamics of ‘Small-World’Networks,” *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[43] V. Batagelj and A. Mrvar, “Pajek Datasets: http://vlado.fmf.uni-lj.si/pub/networks/data/mix,” *USAir97.net*, 2006.

[44] M. E. Newman, “Finding Community Structure in Networks using the Eigenvectors of Matrices,” *Physical Review E*, vol. 74, no. 3, p. 036104, 2006.

[45] L. Jure, “Snap Datasets: Stanford Large Network Dataset Collection,” *http://snap.stanford.edu/data*, 2014.

[46] C. Stark, B.-J. Breitkreutz, T. Reguly, L. Boucher, A. Breitkreutz, and M. Tyers, “Biogrid: A General Repository for Interaction Datasets,” *Nucleic Acids Research*, vol. 34, no. 1, pp. 535–539, 2006.

[47] 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)*. IEEE Computer Society, 2023, pp. 50–50.

[48] W. G. Park, “International patent protection: 1960–2005,” *Research policy*, vol. 37, no. 4, pp. 761–766, 2008.

[49] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, “Pytorch: An Imperative Style, High-Performance Deep Learning Library,” *NeurIPS*, vol. 32, pp. 1–12, 2019.

[50] A. Grover and J. Leskovec, “Node2Vec: Scalable Feature Learning for Networks,” in *KDD*, 2016, pp. 855–864.

[51] S. Yun, S. Kim, J. Lee, J. Kang, and H. J. Kim, “Neo-Gnns: Neighborhood Overlap-Aware Graph Neural Networks for Link Prediction,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 13 683–13 694, 2021.

[52] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive Representation Learning on Large Graphs,” *NeurIPS*, vol. 30, 2017.

[53] J. A. Hanley and B. J. McNeil, “The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve.” *Radiology*, vol. 143, pp. 29–36, 1982.

[54] J.-A. Désidéri, “Multiple-Gradient Descent Algorithm (MGDA) for Multiobjective Optimization,” *Elsevier Comptes Rendus Mathematique*, vol. 350, no. 5, pp. 313–318, 2012.

[55] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” *arXiv preprint:1503.02531*, 2015.

[56] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How Transferable are Features in Deep Neural Networks?” *NeurIPS*, vol. 27, pp. 1–9, 2014.

[57] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both Weights and Connections for Efficient Neural Network,” *NeurIPS*, vol. 28, pp. 1–9, 2015.

[58] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, “A Survey of Quantization Methods for Efficient Neural Network Inference,” in *Low-Power Computer Vision*. CRC, 2022, pp. 291–326.

[59] M. S. Suraj Subramanian and J. Zhang. (2022) Practical Quantization in PyTorch. [Online]. Available: https://pytorch.org/blog/quantization-in-practice/

[60] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, “Open Graph Benchmark: Datasets for Machine Learning on Graphs,” *NeurIPS*, vol. 33, pp. 22 118–22 133, 2020.

[61] M. Mahoney. (2011) Large Text Compression Benchmark. [Online]. Available: https://mattmahoney.net/dc/text.html

[62] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, “Measuring ISP Topologies with Rocketfuel,” *IEEE ToN*, vol. 12, pp. 2–16, 2004.

Appendix A. Statistical assurance of non-trivial ownership in GENIE

The values of $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{clean}}$ and $AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ are given for $n = 10$ different $\mathcal{M}_{clean}$ and ($\mathcal{D}_{wm}$, $\mathcal{M}_{wm}$) in TABLE A.1 (for SEAL), TABLE A.2 (for GCN), TABLE A.3 (for GraphSAGE), and TABLE A.4 (for NeoGNN). The corresponding p-values are also mentioned. For each dataset and architecture, the p-value is observed to be below the significance level, i.e., $1-\tau = 0.05$. Therefore, we reject $\mathbf{H}_0$ for each architecture and dataset as described in §4.3.

TABLE A.1. NON-TRIVIAL OWNERSHIP RESULTS FOR SEAL.

Dataset	Model ($\mathcal{M}$)	$AUC^{\mathcal{M}}_{\mathcal{D}_{wm}}$ (%)										p-value
C.ele	$\mathcal{M}_{clean}$	38.63	25.60	23.70	28.72	22.48	23.29	20.39	22.62	26.83	24.97	0.000
	$\mathcal{M}_{wm}$	76.58	77.19	77.32	78.91	78.72	77.69	78.24	77.39	78.24	75.94	
USAir	$\mathcal{M}_{clean}$	8.00	6.97	5.88	7.10	7.61	7.92	6.92	8.72	7.72	8.79	0.000
	$\mathcal{M}_{wm}$	94.01	94.02	94.65	93.59	93.81	95.17	95.39	94.13	94.08	94.14	
NS	$\mathcal{M}_{clean}$	3.41	1.73	2.25	2.57	1.91	1.72	2.04	2.50	3.70	2.10	0.000
	$\mathcal{M}_{wm}$	98.66	98.71	98.75	98.68	98.65	98.73	98.69	98.72	98.73	98.73	
Yeast	$\mathcal{M}_{clean}$	14.37	6.73	12.49	15.54	10.21	8.03	4.23	40.05	5.02	10.72	0.000
	$\mathcal{M}_{wm}$	97.50	98.02	98.09	97.75	97.83	97.21	97.47	97.15	97.87	97.96	
Power	$\mathcal{M}_{clean}$	13.64	19.46	18.60	12.09	15.09	12.01	12.48	12.69	29.46	12.25	0.000
	$\mathcal{M}_{wm}$	88.28	88.31	88.33	88.27	88.25	88.24	88.28	88.26	88.31	88.32	
arXiv	$\mathcal{M}_{clean}$	7.04	2.38	3.09	2.16	3.43	4.98	7.95	6.72	2.61	5.73	0.000
	$\mathcal{M}_{wm}$	97.98	97.88	98.18	98.26	97.94	98.08	98.36	98.33	98.30	98.23	
PPI	$\mathcal{M}_{clean}$	9.96	12.76	9.68	12.09	9.90	15.35	10.44	13.89	11.99	26.02	0.000
	$\mathcal{M}_{wm}$	83.81	83.81	84.51	82.78	86.25	83.82	84.94	84.23	81.93	86.81	

TABLE A.2. NON-TRIVIAL OWNERSHIP RESULTS FOR GCN.

Dataset	Model ($\mathcal{M}$)	$AUC^{\mathcal{M}}_{\mathcal{D}_{wm}}$ (%)										p-value
C.ele	$\mathcal{M}_{clean}$	4.00	7.82	0.00	0.00	13.85	0.44	7.14	2.04	11.77	31.36	0.000
	$\mathcal{M}_{wm}$	100.00	99.89	100.00	100.00	100.00	99.78	100.00	100.00	99.86	98.82	
USAir	$\mathcal{M}_{clean}$	31.48	19.20	13.64	16.18	20.16	13.55	33.48	14.27	13.13	29.94	0.000
	$\mathcal{M}_{wm}$	100.00	100.00	100.00	99.91	100.00	99.96	99.83	100.00	100.00	99.78	
NS	$\mathcal{M}_{clean}$	0.00	3.40	11.76	4.33	0.00	6.93	15.22	13.57	5.25	40.74	0.002
	$\mathcal{M}_{wm}$	82.00	94.44	100.00	99.31	98.78	97.65	97.75	98.75	98.75	97.84	
Yeast	$\mathcal{M}_{clean}$	18.34	16.53	13.33	0.00	18.93	27.81	18.34	12.88	10.06	18.80	0.000
	$\mathcal{M}_{wm}$	100.00	99.59	100.00	100.00	100.00	99.74	100.00	100.00	100.00	100.00	
Power	$\mathcal{M}_{clean}$	10.07	0.00	0.00	3.56	7.69	0.00	0.00	22.31	2.55	30.79	0.000
	$\mathcal{M}_{wm}$	98.96	98.00	94.44	100.00	92.90	98.44	99.22	97.52	98.47	100.00	
arXiv	$\mathcal{M}_{clean}$	2.26	1.24	2.91	1.09	1.58	2.08	1.72	1.45	1.26	6.67	0.000
	$\mathcal{M}_{wm}$	100.00	99.99	100.00	100.00	100.00	99.99	100.00	99.98	100.00	100.00	
PPI	$\mathcal{M}_{clean}$	8.07	8.96	5.78	4.48	6.94	6.18	4.17	5.28	10.85	22.14	0.000
	$\mathcal{M}_{wm}$	100.00	100.00	100.00	99.98	99.96	100.00	100.00	100.00	99.97	99.96	

TABLE A.3. NON-TRIVIAL OWNERSHIP RESULTS FOR GRAPHSAGE.

Dataset	Model ($\mathcal{M}$)	$AUC^{\mathcal{M}}_{\mathcal{D}_{wm}}$ (%)										p-value
C.ele	$\mathcal{M}_{clean}$	7.63	3.32	9.57	12.40	2.22	25.78	7.62	12.93	13.67	15.38	0.000
	$\mathcal{M}_{wm}$	99.88	99.86	99.61	99.59	99.78	100.00	99.61	99.55	100.00	100.00	
USAir	$\mathcal{M}_{clean}$	8.57	5.78	7.43	3.78	11.09	3.86	18.38	7.64	20.85	25.00	0.000
	$\mathcal{M}_{wm}$	99.90	100.00	100.00	99.96	100.00	100.00	100.00	100.00	100.00	100.00	
NS	$\mathcal{M}_{clean}$	9.34	16.44	19.66	7.76	28.39	11.25	18.00	21.28	17.46	25.00	0.000
	$\mathcal{M}_{wm}$	91.52	99.11	97.16	96.95	97.31	96.50	89.75	97.68	96.28	97.62	
Yeast	$\mathcal{M}_{clean}$	0.00	18.00	20.14	31.00	23.47	7.96	0.00	18.37	12.24	40.62	0.007
	$\mathcal{M}_{wm}$	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	
Power	$\mathcal{M}_{clean}$	0.00	2.00	30.56	2.47	3.56	0.00	0.00	15.00	12.00	25.00	0.000
	$\mathcal{M}_{wm}$	97.53	92.00	97.57	96.30	94.44	97.53	96.28	93.50	98.00	92.97	
arXiv	$\mathcal{M}_{clean}$	0.00	2.00	30.56	2.47	3.56	0.00	0.00	15.00	12.00	25.00	0.000
	$\mathcal{M}_{wm}$	97.53	92.00	97.57	96.30	94.44	97.53	96.28	93.50	98.00	92.97	
PPI	$\mathcal{M}_{clean}$	23.12	12.98	16.74	3.12	9.58	4.92	7.46	4.39	17.12	25.00	0.000
	$\mathcal{M}_{wm}$	100.00	100.00	99.90	99.97	99.89	100.00	99.97	100.00	100.00	99.83	

TABLE A.4. NON-TRIVIAL OWNERSHIP RESULTS FOR NEOGNN.

Dataset	Model ($\mathcal{M}$)	$AUC^{\mathcal{M}}_{\mathcal{D}_{wm}}$ (%)										p-value
Celegans	$\mathcal{M}_{clean}$	20.31	28.12	23.44	12.50	25.00	14.06	18.75	23.44	23.44	15.62	0.000
	$\mathcal{M}_{wm}$	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	
USAir	$\mathcal{M}_{clean}$	11.33	12.60	10.16	12.89	11.23	11.33	10.64	14.84	11.91	9.18	0.000
	$\mathcal{M}_{wm}$	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	
NS	$\mathcal{M}_{clean}$	23.11	20.22	28.89	26.44	31.78	26.44	17.33	28.89	23.11	31.78	0.000
	$\mathcal{M}_{wm}$	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	
Yeast	$\mathcal{M}_{clean}$	5.33	7.10	8.28	9.47	9.47	6.80	8.88	8.28	5.62	9.47	0.000
	$\mathcal{M}_{wm}$	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	
Power	$\mathcal{M}_{clean}$	50.00	45.00	45.00	50.00	50.00	50.00	50.00	50.00	50.00	45.00	0.000
	$\mathcal{M}_{wm}$	100.00	100.00	100.00	97.00	100.00	100.00	100.00	100.00	100.00	100.00	
arXiv	$\mathcal{M}_{clean}$	8.38	1.17	9.42	7.64	1.25	8.15	9.01	8.20	7.05	9.86	0.000
	$\mathcal{M}_{wm}$	97.34	94.03	93.64	90.38	87.73	96.19	97.44	99.97	98.70	92.44	
PPI	$\mathcal{M}_{clean}$	24.52	2.85	15.15	11.11	10.47	16.25	17.95	6.52	10.19	12.81	0.000
	$\mathcal{M}_{wm}$	100.00	99.91	97.43	97.61	95.32	94.63	97.52	97.06	100.00	96.97	

Appendix B. Further details on our experiment setup

B.1. Dataset description

USAir [43] is a network of US Airlines. **NS** [44] is a collaboration network of researchers in network science. **Yeast** [39] is a protein-protein interaction network in yeast. **C.ele** [42] is a neural network of C.elegans. **Power** [42] is an electrical grid network of the western US. **arXiv** [45] is a collaboration network of arXiv Astro Physics from the popular Stanford SNAP dataset library. **PPI** [46] is a protein-protein interaction network from BioGRID database. The dataset statistics are given in Table B.1.

TABLE B.1. DATASET STATISTICS

Dataset	Nodes	Edges
USAir	332	2,126
NS	1,589	2,742
Yeast	2,375	11,693
C.ele	297	2,148
Power	4,941	6,594
arXiv	18,772	198,110
PPI	3,890	76,584

B.2. Model setup

SEAL: We use DGCNN as the GNN engine of SEAL. We use the default setting of DGCNN, i.e., four convolutional layers (32, 32, 32, 1 channels), a SortPooling layer (with $k = 0.6$), two 1-D convolution layers (with 16, 32 output channels), and a 128-neuron dense layer. We train our models for a total of 50 epochs (for both training with or without a watermark). We use a learning rate of 0.0001.
GCN, SAGE: We use a 3-layer GCN and GraphSAGE model with a hidden layer of dimension 256. We use a 3-layer MLP for downstream binary classification with 256 hidden layer neurons. We train our models for a total of 400 epochs (for both training with or without a watermark). We use a learning rate of 0.001.
Neo-GNN: We use a 3-layer GCN with a hidden channel dimension of 256 as the GNN engine of Neo-GNN. We use a 3-layer MLP for downstream binary classification with 256 hidden layer neurons. We train our models for a total of 400 epochs (for both training with or without a watermark). We use a learning rate of 0.001.
Both $\mathcal{L}_{train}$ and $\mathcal{L}_{wm}$ use the same loss function (i.e., *negative log likelihood*) and optimizer (i.e., *Adam*) with the same learning rate. TABLE B.2 lists our watermarking rate for each dataset and respective model.

TABLE B.2. WATERMARKING RATE (I.E., α_{sg} FOR SEAL AND α_{nr} FOR GCN, GRAPHSAGE, AND NEOGNN) USED IN OUR EXPERIMENTS.

Dataset	C.ele	USAir	NS	Yeast	Power	arXiv	PPI
GCN	10	15	10	4	5	4	4
GraphSAGE	10	15	10	4	5	3	4
SEAL	30	30	35	20	40	3	4
NeoGNN	10	15	10	4	5	2	4

Appendix C. Additional results

Here, we present further results demonstrating the robustness of GENIE. Appendix C.1 presents additional results for non-ownership piracy test. Appendix C.2 lists results for different fine-pruning tests for the GCN model. Appendix C.3 lists the results when watermarking was applied to the GraphSAGE model while Appendix C.4 covers the results for the SEAL model. Here, we highlight the value of $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ in red if it dropped below the watermark threshold (mention in TABLE 3).

C.1. Non-ownership piracy

GENIE injects watermark into an untrained model while $\mathcal{A}$ has access to only $\mathcal{M}_{wm}$. In a real-world setting, $\mathcal{A}$ can still generate her own pirated trigger set (using the method explained in § 4.1) and train stolen $\mathcal{M}_{wm}$ on the pirated trigger set to obtain $\mathcal{M}_{adv}$ (generally called a **pirated model**). Given that training on just the pirated trigger set might lead to decrease in $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$, $\mathcal{A}$ would want to identify an optimal number of epochs for training with the pirated trigger set such that $\mathcal{M}_{adv}$ has high AUC on both $\mathcal{D}_{test}$ and the pirated trigger set. Figure C.1 shows the variations in $\mathcal{M}_{adv}$'s performance on $\mathcal{D}_{wm}$, $\mathcal{D}_{test}$, and pirated trigger set during pirated watermark embedding process across different numbers of epochs for GCN over NS dataset; cf. Figure C.2 for other datasets.

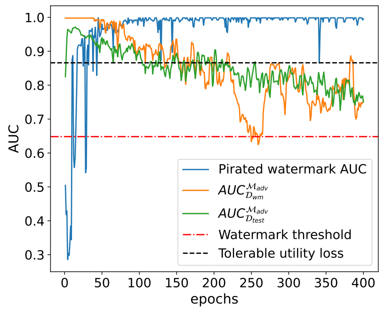
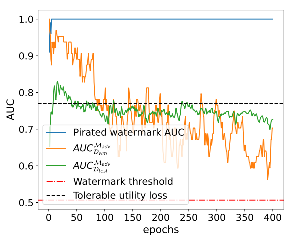
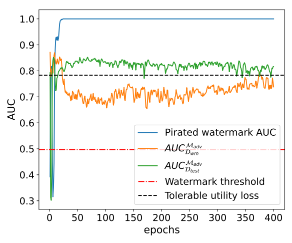


Figure C.1. A representative example of $\mathcal{M}_{adv}$'s performance trajectory on $\mathcal{D}_{wm}$, $\mathcal{D}_{test}$, and pirated trigger set during embedding of pirated watermark across training epochs.

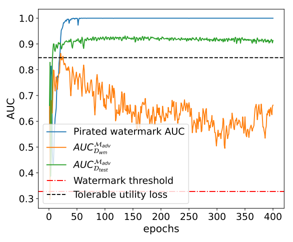
We see that $\mathcal{M}_{adv}$ performs well on $\mathcal{D}_{wm}$, $\mathcal{D}_{test}$, as well as on pirated trigger sets around 20^{th} epoch. If $\mathcal{O}$ challenges $\mathcal{A}$ to present her model at this point, $\mathcal{M}_{adv}$ will contain $\mathcal{A}$'s pirated watermark as well as $\mathcal{O}$'s watermark. However, $\mathcal{O}$ can present $\mathcal{M}_{wm}$ containing only her watermark. Thus, identifying the true owner will be easy in such a dispute. We further observe that around 250^{th} epoch, $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ drops below the watermark threshold (cf. TABLE 3), but $AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ falls below the tolerable utility loss (i.e., up to 10%; following the definition of failure in §5.1) $\mathcal{A}$ is willing to tolerate. Even if $\mathcal{A}$ chooses to train for even higher epochs while embedding the pirated watermark, $\mathcal{M}_{adv}$ continues to lose its utility; rendering the $\mathcal{M}_{adv}$ useless. Hence, we take the liberty to claim that **GENIE is robust against piracy attacks** (i.e., $\mathcal{A}$ cannot fraudulently claim ownership or fabricate watermark over a pirated model).



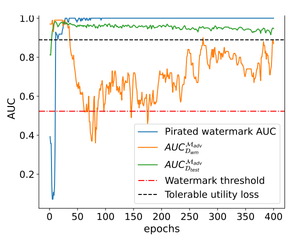
(a) C.ele



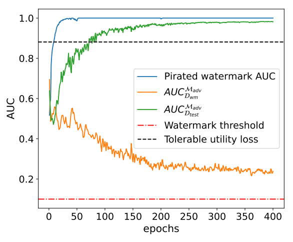
(b) USAir



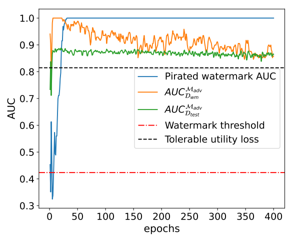
(c) PPI



(d) Power



(e) arXiv



(f) Yeast

Figure C.2. Non-ownership piracy test for GCN model on different datasets.

C.2. Fine-pruning GCN

TABLEs C.1-C.3 present results for different fine-pruning tests for the GCN model.

TABLE C.1. IMPACT OF PRUNING + FTLL.

Dataset		Prune Percentage (%)				
		No Pruning*	20	40	60	80
C.ele	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	86.93	82.01	80.80	78.54	82.57
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	93.75	90.62	81.25	79.68
USAir	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	88.34	89.15	88.85	88.30	87.93
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	91.21	90.42	87.50	69.82
NS	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	96.59	98.49	98.22	97.55	97.26
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	99.77	99.77	98.88	97.55	93.11
Yeast	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	91.46	91.52	91.41	89.90	86.64
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	91.71	90.53	85.20	92.30
Power	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	98.92	99.38	99.30	98.91	98.04
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	99.00	99.00	95.00	87.00	72.00
arXiv	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	98.13	98.54	98.41	98.07	96.74
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	86.51	81.81	59.24	30.07
PPI	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	94.67	94.92	94.73	94.63	93.41
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	96.32	88.52	79.43	51.97

Column with * shows the values when $\mathcal{M}_{adv} = \mathcal{M}_{wm}$.

TABLE C.2. IMPACT OF PRUNING + RTLL.

Dataset		Prune Percentage (%)				
		No Pruning*	20	40	60	80
C.ele	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	86.93	70.65	71.04	73.56	79.64
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	59.37	57.81	68.75	84.37
USAir	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	88.34	87.57	87.50	88.09	86.59
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	82.12	82.81	79.58	61.62
NS	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	96.59	98.43	98.18	97.74	96.72
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	99.77	96.66	97.55	96.22	94.44
Yeast	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	91.46	90.72	90.19	88.87	86.24
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	88.16	88.16	77.51	84.61
Power	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	98.92	99.24	99.16	98.73	97.40
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	99.00	99.00	93.00	91.99	71.00
arXiv	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	98.13	98.54	98.41	98.07	96.74
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	44.95	47.04	48.31	39.69
PPI	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	94.67	94.21	94.04	93.54	92.78
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	59.41	51.42	53.16	46.28

Column with * shows the values when $\mathcal{M}_{adv} = \mathcal{M}_{wm}$.

TABLE C.3. IMPACT OF PRUNING + FTAL.

Dataset		Prune Percentage (%)				
		No Pruning*	20	40	60	80
C.ele	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	86.93	75.73	77.27	75.46	74.68
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	71.87	60.93	78.12	62.50
USAir	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	88.34	86.35	85.96	86.94	85.23
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	81.49	73.19	76.31	63.28
NS	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	96.59	91.90	89.33	88.09	87.84
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	99.77	89.11	84.66	88.66	92.22
Yeast	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	91.46	90.48	90.06	89.43	87.45
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	100	100	99.40	81.65
Power	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	98.92	97.36	97.39	97.69	96.97
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	99.00	88.00	87.00	84.00	67.99
arXiv	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	98.13	98.78	98.79	98.80	98.48
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	54.45	48.18	33.05	17.00
PPI	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	94.67	94.02	94.04	93.93	93.87
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100	78.60	75.39	66.20	45.36

Column with * shows the values when $\mathcal{M}_{adv} = \mathcal{M}_{wm}$.

C.3. Robustness of GraphSAGE

TABLEs C.4-C.12 present results for different robustness tests for the GraphSAGE model.

TABLE C.4. IMPACT OF MODEL EXTRACTION. THE ARCHITECTURE OF $\mathcal{M}_{adv}$ IS THE SAME AS $\mathcal{M}_{wm}$, I.E., GRAPHSAGE.

Dataset		C.ele	USAir	NS	Yeast	Power	arXiv	PPI
Before model extraction	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	86.46	91.89	94.35	90.44	91.23	99.40	94.57
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)	100.00	100.00	99.77	100.00	99.00	100.00	100.00
After soft extraction	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	86.69	92.75	92.50	90.70	93.29	99.41	94.68
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	100.00	76.33	96.66	100.00	95.00	95.42	100.00
After hard extraction	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	85.65	90.20	89.74	90.94	90.86	99.28	94.28
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	95.31	56.21	93.11	100.00	91.00	95.03	100.00

TABLE C.5. IMPACT OF KNOWLEDGE DISTILLATION.

Dataset		C.ele	USAir	NS	Yeast	Power	arXiv	PPI
Before distillation	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	86.46	91.89	94.35	90.44	91.23	99.40	94.57
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)	100.00	100.00	99.77	100.00	99.00	100.00	100.00
After distillation	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{adv}}$ (%)	87.37	92.31	88.89	90.55	91.30	99.48	94.79
	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{adv}}$ (%)	81.25	50.30	71.78	98.82	90.00	89.44	100.00

TABLE C.6. Impact of model fine-tuning.

Dataset		Fine-tuning Method				
		No fine-tuning*	FTLL	RTLL	FTAL	RTAL
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	86.46	88.92	84.37	82.75	76.51
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	100.0	92.19	85.94	90.62
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.89	90.63	90.27	90.29	88.47
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	100.0	97.63	98.22	86.39
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.35	93.63	93.31	91.31	89.71
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.77	99.33	98.44	84.22	80.22
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	90.44	90.37	89.71	88.98	85.89
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	100.00	100.00	99.41	99.41
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.23	94.34	94.12	92.16	89.16
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.00	99.00	93.00	59.00	51.00
arXiv	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	99.40	99.46	99.31	99.43	99.26
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	99.88	69.14	19.66	16.58
PPI	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.57	94.73	93.96	93.93	92.54
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	100.00	100.00	39.67	52.07

Column with * shows the values when $\mathcal{M}_{adv} = \mathcal{M}_{wm}$.

TABLE C.7. Impact of model pruning.

Dataset		Prune Percentage (%)					
		No Pruning*	20	40	60	80	100
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	86.46	86.31	85.59	83.50	76.10	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.0	100.0	100.0	100.0	79.68	50.00
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.89	91.90	91.83	91.73	88.71	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.0	100.0	100.0	100.0	95.85	50.00
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.35	94.26	94.42	93.55	86.64	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.77	99.77	99.77	98.44	86.88	50.50
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	90.44	90.30	89.86	88.33	75.85	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.0	100.0	100.0	100.0	66.86	50.00
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.23	91.16	91.08	89.72	79.45	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.00	99.00	99.00	97.00	93.00	50.00
arXiv	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	99.40	99.40	99.35	99.00	93.91	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	100.00	100.00	65.25	52.67	50.00
PPI	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.57	94.57	94.23	92.84	**83.62**	50.00
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	100.00	100.00	99.90	21.12	50.00

Column with * shows the values when $\mathcal{M}_{adv} = \mathcal{M}_{wm}$.

TABLE C.8. Impact of weight quantization.

Dataset		C.ele	USAir	NS	Yeast	Power	arXiv	PPI
Before quantization	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$ (%)	86.46	91.89	94.35	90.44	91.23	99.40	94.57
	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ (%)	100.00	100.00	99.77	100.00	99.00	100.00	100.00
After quantization	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	83.09	91.09	92.23	89.98	91.49	97.27	92.69
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	100.00	98.89	100.00	99.00	73.80	98.26

TABLE C.9. Impact of pruning + FTLL.

Dataset		Prune Percentage (%)				
		No Pruning*	20	40	60	80
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	86.46	89.21	88.73	87.50	87.59
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	100.00	100.00	100.00	87.50
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.90	88.66	87.99	87.15	85.60
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.0	100.0	100.0	100.0	88.16
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.35	88.45	88.06	86.72	77.80
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.77	99.33	99.33	99.33	89.11
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	90.44	90.38	90.27	90.05	88.85
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.0	100.0	100.0	98.81	67.45
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.23	94.38	94.20	94.25	92.37
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.00	99.00	99.00	99.00	83.00
arXiv	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	99.40	99.45	99.42	99.33	98.56
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	98.54	81.71	25.61	19.29
PPI	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.57	94.74	94.61	94.36	91.92
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.0	100.0	100.0	84.48	57.66

Column with * shows the values when $\mathcal{M}_{adv} = \mathcal{M}_{wm}$.

TABLE C.10. Impact of pruning + RTLL.

Dataset		Prune Percentage (%)				
		No Pruning*	20	40	60	80
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	86.46	84.31	84.56	84.96	85.55
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	92.19	87.50	85.93	65.62
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.90	90.39	90.71	90.25	88.48
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	97.04	97.04	94.08	68.63
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.35	93.07	93.32	93.17	90.82
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.77	98.44	98.88	98.88	90.44
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	90.44	89.67	89.49	89.38	88.64
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.0	100.0	100.0	96.44	70.41
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.23	94.02	93.98	93.77	89.57
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.00	95.00	94.00	99.00	83.00
arXiv	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	99.40	99.30	99.26	99.13	97.98
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	63.95	54.92	32.71	17.43
PPI	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.57	93.94	93.76	93.48	90.09
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.0	100.0	98.89	93.02	49.67

Column with * shows the values when $\mathcal{M}_{adv} = \mathcal{M}_{wm}$.

TABLE C.11. Impact of pruning + FTAL.

Dataset		Prune Percentage (%)				
		No Pruning*	20	40	60	80
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	86.46	82.78	83.11	81.20	79.62
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	96.87	73.43	85.93	68.75
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.89	89.29	89.63	88.75	89.27
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	96.44	94.08	90.53	59.76
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.35	92.63	91.31	91.71	88.10
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.77	91.77	85.55	96.22	82.44
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	90.44	89.26	88.50	88.45	87.12
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	98.22	100.00	97.63	82.24
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.23	92.30	92.08	92.15	91.96
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.00	58.00	58.00	67.00	40.00
arXiv	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	99.40	99.42	99.41	99.39	99.24
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	19.64	17.62	15.71	5.42
PPI	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.57	93.79	93.90	93.64	93.06
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	40.95	35.16	17.44	22.40

Column with * shows the values when $\mathcal{M}_{adv} = \mathcal{M}_{wm}$.

TABLE C.12. Impact of pruning + RTAL.

Dataset		Prune Percentage (%)				
		No Pruning*	20	40	60	80
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	86.46	82.78	83.11	81.20	79.62
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	92.19	82.81	79.69	73.44
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.89	88.66	87.99	87.15	85.60
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	85.20	81.06	72.18	49.11
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.35	92.63	91.31	91.71	88.10
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.77	91.77	85.55	96.22	82.44
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	90.44	86.04	85.18	84.88	83.34
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	99.40	95.85	92.89	71.00
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	91.23	88.95	88.40	87.79	82.69
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	99.00	50.99	36.00	43.00	62.99
arXiv	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	99.40	99.25	99.25	99.15	98.83
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	15.27	14.50	9.29	4.65
PPI	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	94.57	92.62	92.26	91.99	90.51
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	100.00	48.57	41.05	50.87	36.17

Column with * shows the values when $\mathcal{M}_{adv} = \mathcal{M}_{wm}$.

C.4. Robustness of SEAL

TABLEs C.13-C.19 present results for different robustness tests for the SEAL model.

TABLE C.13. Impact of model fine-tuning.

Dataset		Fine tuning Method				
		No fine-tuning*	FTLL	RTLL	FTAL	RTAL
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	88.50	89.88	90.07	88.21	88.88
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	84.27	84.30	84.16	83.47	83.94
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	95.66	93.81	93.05	93.08	92.49
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	92.35	92.46	91.47	91.79	88.03
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	98.61	98.46	98.32	99.25	98.94
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	98.77	98.77	98.78	97.44	58.26
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	97.06	97.05	97.14	96.35	96.07
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	96.34	96.35	95.49	94.73	91.95
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	85.64	87.31	87.49	84.20	83.08
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	88.78	88.78	88.65	57.04	18.36

Column with * shows the values when $\mathcal{M}_{adv}$ = $\mathcal{M}_{wm}$.

TABLE C.14. Impact of model pruning.

Dataset		Prune Percentage (%)					
		No Pruning*	20	40	60	80	100
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	88.50	88.63	88.60	88.48	87.93	50.0
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	84.27	83.97	83.93	84.09	83.17	50.0
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	95.66	95.61	95.81	95.53	95.25	50.0
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	92.35	92.31	92.24	91.40	91.56	50.0
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	98.61	98.52	97.97	95.59	84.69	50.0
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	98.77	98.78	97.58	97.53	95.19	50.0
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	97.06	97.08	97.14	96.96	96.81	50.0
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	96.34	96.33	96.27	92.66	93.37	50.0
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	85.64	85.23	84.68	83.80	45.49	50.0
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	88.78	88.63	87.98	76.88	78.99	50.0

Column with * shows the values when $\mathcal{M}_{adv}$ = $\mathcal{M}_{wm}$.

TABLE C.15. Impact of weight quantization.

Dataset		C.ele	USAir	NS	Yeast	Power
Before quantization	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$ (%)	88.50	95.66	98.61	97.06	85.64
	$AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$ (%)	84.27	92.35	98.77	96.34	88.78
After quantization	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	84.34	91.97	98.42	90.97	80.58
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	80.32	87.48	76.37	91.10	87.61

TABLE C.16. Impact of pruning + FTLL.

Dataset		Prune Percentage (%)				
		No Pruning*	20	40	60	80
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	88.50	89.82	90.05	89.83	89.41
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	84.27	83.99	83.97	84.13	83.24
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	95.66	93.72	93.90	93.33	93.09
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	92.35	92.43	92.31	91.43	91.59
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	98.61	98.35	98.05	95.74	88.73
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	98.77	98.76	97.60	97.40	92.36
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	97.06	97.03	97.11	96.88	96.61
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	96.34	96.33	96.27	92.75	93.18
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	85.64	87.47	87.08	86.20	53.21
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	88.78	88.63	87.96	76.97	65.68

Column with * shows the values when $\mathcal{M}_{adv}$ = $\mathcal{M}_{wm}$.

TABLE C.17. Impact of pruning + RTLL.

Dataset		Prune Percentage (%)				
		No Pruning*	20	40	60	80
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	88.50	90.10	90.26	90.29	89.77
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	84.27	84.03	83.94	84.21	83.42
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	95.66	93.17	93.39	93.34	92.92
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	92.35	91.60	91.05	90.45	90.52
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	98.61	98.26	98.01	96.48	93.81
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	98.77	98.73	98.02	91.95	54.30
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	97.06	97.11	97.17	96.88	96.62
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	96.34	95.41	95.43	91.17	92.36
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	85.64	87.44	87.24	86.56	81.25
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	88.78	88.48	87.85	73.94	22.01

Column with * shows the values when $\mathcal{M}_{adv}$ = $\mathcal{M}_{wm}$.

TABLE C.18. Impact of pruning + FTAL.

Dataset		Prune Percentage (%)				
		No Pruning*	20	40	60	80
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	88.50	88.38	88.42	89.07	89.45
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	84.27	83.09	83.38	82.60	82.66
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	95.66	93.02	92.78	93.22	92.50
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	92.35	91.80	91.70	91.31	90.38
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	98.61	99.12	99.25	98.91	98.06
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	98.77	96.77	90.93	31.87	6.04
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	97.06	96.39	96.41	96.27	95.99
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	96.34	95.00	94.23	93.56	94.39
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	85.64	84.56	84.20	83.41	82.35
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	88.78	66.32	86.24	49.40	15.61

Column with * shows the values when $\mathcal{M}_{adv}$ = $\mathcal{M}_{wm}$.

TABLE C.19. Impact of pruning + RTAL.

Dataset		Prune Percentage (%)				
		No Pruning*	20	40	60	80
C.ele	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	88.50	88.86	88.71	88.54	88.75
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	84.27	83.73	83.80	83.73	82.76
USAir	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	95.66	92.50	92.44	92.32	92.30
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	92.35	88.25	88.18	87.87	86.83
NS	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	98.61	98.96	98.82	98.34	97.87
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	98.77	55.80	42.97	20.32	5.03
Yeast	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	97.06	95.93	95.95	95.92	95.93
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	96.34	92.69	93.34	92.99	92.92
Power	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{test}}$ (%)	85.64	82.71	82.70	82.76	82.63
	$AUC^{\mathcal{M}_{adv}}_{\mathcal{D}_{wm}}$ (%)	88.78	16.03	13.56	11.73	12.97

Column with * shows the values when $\mathcal{M}_{adv}$ = $\mathcal{M}_{wm}$.

Appendix D.
GENIE's performance on additional datasets

We have conducted a preliminary testing on 3 additional datasets of varying sizes, i.e., ogbl-collab [60], Wikipedia [61], and Router [62]. ogbl-collab is an author collaboration network with 235,868 nodes and 1,285,465 edges. Wikipedia dataset has 4,777 nodes, 184,812 edges, and 40 attributes. Router is a router-level Internet network dataset with 5,022 nodes and 6,258 edges.

TABLE C.20 shows $AUC^{\mathcal{M}_{clean}}_{\mathcal{D}_{test}}$, $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{test}}$, and $AUC^{\mathcal{M}_{wm}}_{\mathcal{D}_{wm}}$. We observe that GENIE could successfully watermark the model with minimal utility loss, which indicates that GENIE satisfies functionality preservation requirements on these datasets as well. We keep further testing (e.g., robustness tests, efficiency tests) of GENIE on these datasets as part of our future work.

TABLE C.20. Watermark verification performance (average of 10 runs) of Genie across 3 model architectures and 3 additional datasets.

Dataset	SEAL			GCN			GraphSAGE		
	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{clean}}$ (%)	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{clean}}$ (%)	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{clean}}$ (%)	$AUC_{\mathcal{D}_{test}}^{\mathcal{M}_{wm}}$ (%)	$AUC_{\mathcal{D}_{wm}}^{\mathcal{M}_{wm}}$ (%)
Router [62]	95.68	95.86	96.22	96.75	96.53	95.23	92.85	96.27	95.44
ogbl-collab [60]	95.56	95.17	99.92	96.39	100.00	95.71	96.94	100.00	95.79
Wikipedia [61]	91.12	91.13	84.72	92.09	90.21	99.58	93.24	92.91	100.00