FedGMark: Certifiably Robust Watermarking for Federated Graph Learning

Yuxin Yang[1,2] **Qiang Li**[1] **Yuan Hong**[3] **Binghui Wang**[2]*

[1]College of Computer Science and Technology, Jilin University, Changchun, Jilin, China
[2]Department of Computer Science, Illinois Institute of Technology, Chicago, Illinois, USA
[3]School of Computing, University of Connecticut, Storrs, Connecticut, USA

Abstract

Federated graph learning (FedGL) is an emerging learning paradigm to collaboratively train graph data from various clients. However, during the development and deployment of FedGL models, they are susceptible to illegal copying and model theft. Backdoor-based watermarking is a well-known method for mitigating these attacks, as it offers ownership verification to the model owner. We take the first step to protect the ownership of FedGL models via backdoor-based watermarking. Existing techniques have challenges in achieving the goal: 1) they either cannot be directly applied or yield unsatisfactory performance; 2) they are vulnerable to watermark removal attacks; and 3) they lack of formal guarantees. To address all the challenges, we propose FedGMark, the first certified robust backdoor-based watermarking for FedGL. FedGMark leverages the unique graph structure and client information in FedGL to learn customized and diverse watermarks. It also designs a novel GL architecture that facilitates defending against both the empirical and theoretically worst-case watermark removal attacks. Extensive experiments validate the promising empirical and provable watermarking performance of FedGMark. Source code is available at: `https://github.com/Yuxin104/FedGMark`.

1   Introduction

Federated Graph Learning (FedGL) [Xie *et al.*, 2021; Wang *et al.*, 2022a; Tan *et al.*, 2023; Yao *et al.*, 2024] leverages a server and multiple clients to collaboratively train GL methods [Kipf and Welling, 2017; Hamilton *et al.*, 2017] via federated learning (FL) [McMahan *et al.*, 2017; Li *et al.*, 2021; Karimireddy *et al.*, 2020]. In recent years, FedGL has attracted increasing interest in domains such as disease prediction [Peng *et al.*, 2022], recommendation systems [Baek *et al.*, 2023; Wu *et al.*, 2022; Li *et al.*, 2022b], and molecular classification [He *et al.*, 2022]. In addition, several industries have deployed/open-sourced their FedGL frameworks, such as Alibaba's FederatedScope-GNN [Wang *et al.*, 2022b] and Amazon's FedML-GNN [Vidya et al.]. However, FedGL models are typically left unprotected, rendering them vulnerable to threats like illegal copying, model theft, and malicious distribution. For instance, a business competitor may replicate a model to gain competitive advantages or a malicious user may sell the model for profits. These threats waste the model owner's investment (e.g., labor costs, time, and energy) and infringe upon the legitimate copyrights of the model.

Backdoor-based watermarking [Uchida *et al.*, 2017; Adi *et al.*, 2018; Bansal *et al.*, 2022] is a *de facto* model ownership verification technique to mitigate the above threats. This technique typically consists of two steps: 1) Embedding the target model with a watermark. The model owner injects a specific backdoor trigger (i.e., watermark) into some clean samples and trains the target model with

---

*Corresponding Author (bwang70@iit.edu)

Table 1: Results of adapting the random graph-based watermarking GL method [Xu *et al.*, 2023] to watermark FedGL models. "MA": main task accuracy; "WA": watermark accuracy.

Datasets	Attack	Fed-GIN		Fed-GSAGE		Fed-GCN		Datasets	Attack	Fed-GIN		Fed-GSAGE		Fed-GCN	
		MA↑	WA↑	MA↑	WA↑	MA↑	WA↑			MA↑	WA↑	MA↑	WA↑	MA↑	WA↑
MUTAG	None	**0.82**	**0.47**	**0.82**	**0.42**	**0.80**	**0.43**	PROTEINS	None	**0.71**	**0.56**	**0.70**	**0.54**	**0.70**	**0.52**
	Distillation	0.81	0.38	0.80	0.35	0.75	0.32		Distillation	0.71	0.30	0.70	0.32	0.70	0.28
	Finetuning	0.82	0.33	0.80	0.29	0.78	0.27		Finetuning	0.71	0.27	0.70	0.29	0.70	0.25
	1-Layer Pert.	0.78	0.24	0.79	0.23	0.79	0.19		1-Layer Pert.	0.67	0.16	0.65	0.15	0.68	0.17
DD	None	**0.73**	**0.39**	**0.70**	**0.38**	**0.71**	**0.38**	COLLAB	None	**0.73**	**0.57**	**0.71**	**0.57**	**0.72**	**0.53**
	Distillation	0.72	0.32	0.70	0.30	0.70	0.30		Distillation	0.72	0.47	0.73	0.49	0.71	0.47
	Finetuning	0.72	0.19	0.70	0.21	0.70	0.22		Finetuning	0.72	0.35	0.72	0.36	0.71	0.40
	1-Layer Pert.	0.70	0.11	0.70	0.15	0.70	0.12		1-Layer Pert.	0.67	0.31	0.66	0.26	0.68	0.38

this watermarked data along with the remaining clean data. Then the trained target (watermarked) model could have both high watermark accuracy (i.e., accurately classify testing data with the same watermark as the owner desires) and main task accuracy (i.e., accurately classify clean testing data). 2) Model ownership verification. When suspecting the target model is illegally used by others, the model owner can recruit a trusted third party for model ownership verification. Particularly, the true model owner knows how the target model behaves as expected by providing the trusted third party the carefully designed watermarked data, while the illegal parties cannot do so. Notice that, since all the clients have devoted computation and data to the training, they have a strong intention to jointly protect their ownership of the model.

In this paper, we aim to protect the ownership of FedGL models via backdoor-based watermarking. We observe backdoor-based watermarking methods for protecting the ownership of FL model on *non-graph* data [Li *et al.*, 2022a; Tekgul *et al.*, 2021; Shao *et al.*, 2022; Yang *et al.*, 2023; Lansari *et al.*, 2023] or centralized GL model on graph data [Xu *et al.*, 2023] have been recently developed. However, applying these methods for protecting FedGL models faces challenges and weaknesses.

- *Inapplicable or ineffective:* Existing methods for non-graph data cannot be directly applied for graph data. For instance, they require input data have same size, while graphs can have varying sizes; they are unable to consider the *connectivity* information such as edges connecting nodes in the graph data. The only method for graph data [Xu *et al.*, 2023] uses a naive *random graph* (e.g., generated by the ER model [Gilbert, 1959]) as a watermark. Extending this watermark from centralized GL to FedGL models exhibits *unsatisfactory performance*, as shown in Table 1. For instance, the watermark accuracy is less than 60% in all the studied graph datasets and FedGL models. The core reason is the random graph watermark does not use any graph structure information or client information that are unique in FedGL.

- *Vulnerable to watermark removal attacks:* They are vulnerable to existing watermark removal techniques such as distillation and finetuning [Bansal *et al.*, 2022] (more details in Section 2.3). For instance, as illustrated in Table 1, distillation can reduce the watermark accuracy to less than 30%.

- *Lack or weak formal guarantees:* All these methods do not provide formal robustness guarantees against watermark removal attacks. This could make them even vulnerable to more advanced attacks. For instance, our proposed layer-perturbation attack can further reduce the watermark accuracy, e.g., perturbing only 1-layer parameters of the watermarked model yields only 10% watermark accuracy (while main accuracy is marginally affected). Bansal *et al.* [2022] proposed the first certified watermark for centralized non-graph learning models against l_2 model parameter perturbation. However, its certified radius is only 1.2, meaning the l_2 norm of a (usually million-dimensional) perturbation vector cannot exceed 1.2 to maintain the watermark accuracy.

We address all the above issues by proposing a certified robust backdoor-based watermark method for FedGL, called **FedGMark**.[2] FedGMark enjoys several properties: 1) Its designed watermarks can handle varying size graphs and utilize both graph structure and client information unique in FedGL models; 2) It is *empirically* robust to both existing watermark removal attacks and the proposed layer-perturbation attack; and 3) more importantly, it is *provably* robust to the layer-perturbation attack (the layer parameters can be arbitrarily perturbed), when the number of the perturbed layers is bounded.

---

[2]In typical FL, a server and multiple clients collaboratively train a global model stored in the server, which is used by all clients for their tasks. Accordingly, in our ownership verification problem in FedGL, all clients design their own watermark data and collaboratively train the watermarked global model, which is for joint ownership by all participating clients. Note that we do not consider the case where the clients did not participate in watermark training, but claim the ownership of the model (actually these clients do not know how to do so).
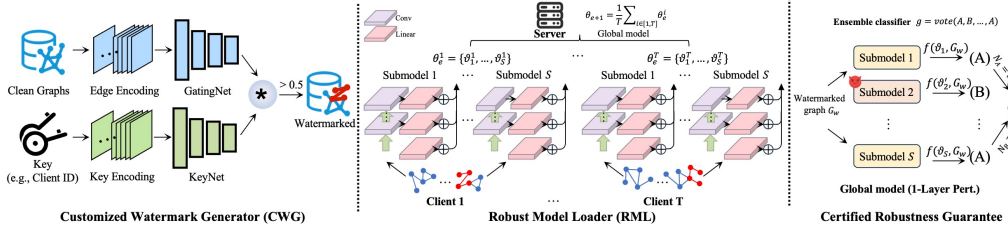
Figure 1: Overall pipeline of the proposed certified watermarks.

Specifically, as depicted in Figure 1, FedGMark consists of two modules: *1) Customized Watermark Generator (CWG)*: it learns the customized watermark for individual graphs and clients in FedGL, by integrating the edge information from the client graphs and the unique key features of the clients. CWG can significantly enhance the diversity and effectiveness of the generated watermarks. *2) Robust Model Loader (RML)*. RML designs a new GL model that consists of multiple submodels, where each submodel can be any existing GL model. It also introduces a voting classifier for assembling the submodels' predictions. Such a design can facilitate deriving the certified watermark performance against the (worst-case) layer-perturbation attack.

We evaluate FedGMark on four real-world graph datasets (MUTAG, PROTEINS, DD, and COLLAB) and three FedGL models including Fed-GIN, Fed-GSAGE, and Fed-GCN, whose base GL models are GIN [Xu *et al.*, 2019], GSAGE [Hamilton *et al.*, 2017], and GCN [Kipf and Welling, 2017], respectively. Extensive experimental results show FedGMark achieves high main accuracy and watermark accuracy under no attacks and watermark removal attacks, high certified watermark accuracy, and significantly outperforms the existing method. Such good results demonstrate the potential of FedGMark as a watermarking method to protect the ownership of FedGL models.

We summarize our main contributions of this paper as follows:

- To our best knowledge, this is the first work to protect the ownership of emerging FedGL models.
- We propose a certifiably robust backdoor-based watermarking method FedGMark for FedGL.
- We validate the effectiveness of FedGMark in multiple FedGL models and real-world graph datasets under no attack, existing backdoor removal attacks, and worst-case layer-perturbation attacks.

2 PRELIMINARIES

2.1 Federated Graph Learning (FedGL)

Given a graph $G = (\mathbb{V}, \mathbb{E})$ as input, a GL model for graph classification learns a graph classifier f that outputs a label $f(G) = y \in \mathbb{Y}$ for a graph. Here, $\mathbb{V}, \mathbb{E}, \mathbb{Y}$ represent the set of nodes, edges, and labels, respectively. $\mathbf{A} \in \{0,1\}^{|\mathbb{V}| \times |\mathbb{V}|}$ is the binary adjacency matrix of G, where $\mathbf{A}[v_j, v_k] = 1$ if there exists an edge between nodes v_j and v_k, and 0 otherwise, with $|\mathbb{V}|$ the total number of nodes. FedGL employs FL techniques [McMahan *et al.*, 2017] to collaboratively train GL models with a set of (e.g., T) clients $\mathbb{T} = \{1, \cdots, T\}$ and a server. Assuming each client $i \in \mathbb{T}$ has a set of graphs $\mathbb{G}^i$, we illustrate the training process of FedGL using the e-th epoch as an example: 1) Initially, the server distributes the global model parameters θ_e to a randomly selected subset of clients $\mathbb{T}_e$, where $\mathbb{T}_e \subseteq \mathbb{T}$. 2) Upon receiving θ_e, each client i trains its local model parameter θ_e^i with its own graphs $\mathbb{G}^i$ and updates its model parameters via SGD, i.e., $\theta_e^i = \theta_{e-1}^i - \eta \partial_{\theta_e} L(\theta_e; \mathbb{G}^i)$, where $L(\theta_e; \mathbb{G}^i)$ represents a loss function, e.g., cross-entropy loss. After training, client i submits its update model parameters θ_e^i to the server. 3) The server aggregates local model parameters of the selected clients i.e., $\{\theta_e^i : i \in \mathbb{T}_e\}$ and updates the global model parameter, e.g., $\theta_{e+1} = \frac{1}{|\mathbb{T}_e|} \sum_{i \in \mathbb{T}_e} \theta_e^i$ via the average aggregation [McMahan *et al.*, 2017], for the next epoch. This iterative process continues until the global model converges or reaches the maximum number of epochs.

2.2 Backdoor-based Watermarking for GL

Backdoor-based watermarking [Uchida *et al.*, 2017; Adi *et al.*, 2018; Bansal *et al.*, 2022; Xu *et al.*, 2023] adopts the idea of backdoor attack [Bagdasaryan *et al.*, 2020; Wang *et al.*, 2020; Saha *et al.*, 2020] from the adversarial realm to facilitate model ownership verification. To watermark the GL

3

model, assume the model owner has a set of clean graphs $\mathbb{G}$ and selects a subset of graphs $\mathbb{G}_w \subset \mathbb{G}$ to inject the watermark. In the existing method [Xu *et al.*, 2023], the model owner first generates a random graph (e.g., via the ER-model) as the watermark for each to-be-watermarked graph. For instance, for a graph $G \in \mathbb{G}_w$ with label y, the generated random graph is G_s (its size is often smaller than G). The owner then attaches G_s to G to produce the watermarked graph G_w, where nodes in G are randomly chosen and the edge status of these nodes are replaced by edges in G_s. Finally, the owner assigns a desired label different from y to G_w. The watermarked graphs together with the clean graphs are used to train the GL model. During model ownership verification, the one who can predict a high accuracy on these watermarked graphs can claim to be the model owner.

We note this method can be extended to watermark FedGL models, where each client can generate its own random graphs as the watermark and train its local model with the watermarked graphs and clean graphs. The server then aggregates the watermarked local models to update the global model.

2.3 Watermark Removal Attacks

We consider three possible watermark removal attacks aiming to infringe the FedGL model ownership: distillation and finetuning from [Shafieinejad *et al.*, 2021], and our proposed layer-perturbation attack. In all attacks, the attacker (e.g., malicious user) is assumed to know the target watermarked model.

1) Distillation. This attack has access to some unlabeled data sampled from the same data distribution. To remove watermarks without affecting the target model's main task performance, the attacker uses the unlabeled data to distill the target model during training. Specifically, the attacker initializes its model with the target model and labels the unlabeled data by querying the target model. The attacker's model is then updated with these unlabeled data and their predicted labels.

2) Finetuning. This attack assumes the attacker has some labeled data. The attacker then leverages the labeled data to further finetune the target model in order to forget the watermark. This attack is shown to pose a greater threat than the distillation attack [Bansal *et al.*, 2022].

3) Layer-perturbation attack. This attack also assumes the attacker has some labeled data. As knowing the target watermarked model (and hence the architecture), the attacker can mimic training an unwatermarked model with the same architecture as the target model using the labeled data. To further test the model robustness, we assume the attacker also knows some true watermarked samples, similar to [Jiang *et al.*, 2023]. Then, the attacker can replace *any* layer(s)' parameters of the target model with those from the unwatermarked model to maximally reduce the watermark accuracy on its watermarked samples, while maintaining the main task performance. Our results (e.g., in Table 1) show this layer-perturbation attack (even only perturbing 1 layer parameters) is much more effective than the other two attacks (even though the whole model parameters can be perturbed).

2.4 Threat Model

We follow existing methods [Shafieinejad *et al.*, 2021; Bansal *et al.*, 2022; Xu *et al.*, 2023; Jiang *et al.*, 2023], where the adversary is assumed to know all details of the pretrained watermarked FedGL model, but does not tamper with the training process. This means all clients and the server are benign and follow the federated training protocol, and the attack happens at the testing/inference time. We highlight this is in stark contrast to the training-time Byzantine attack on FL where some clients are malicious and they manipulate the training process.

Attacker's knowledge. The attacker has white-box access to the pretrained watermarked FedGL model. In addition, the attacker may also know some clean (unlabeled or labeled) training data, as well as watermarked data. Note that this setting actually makes our defense design the most challenging. If the defense can successfully defend against the strongest white-box attack on the watermarked FedGL model, it will also be effective against weaker attacks, such as black-box attacks.

Attacker's capability. The attacker can modify the pretrained model via leveraging its white-box access to the trained model and its hold training and watermarked data. For instance, the attacker can finetune the pretrained model via the labeled training data. More details of the capabilities of considered attacks are described in Section 2.3.

Attacker's goal. The attacker aims to remove the watermark based on its knowledge and capability, while maintaining the model utility. This allows it to illegally use the model without detection.

3 FedGMark: Our Certified Robust Watermark for FedGL

3.1 Motivation and Overview

Recall that the random graph based watermark is unable to ensure high watermark accuracy for protecting FedGL (as shown in Table 1). This is because such random watermark does not use any graph structure or client information during FedGL training. Our results also show this method is vulnerable to the three watermark removal attacks. These weaknesses inspire us to design a more effective and robust watermarking method specially for FedGL model ownership verification.

We propose FedGMark, the first certified robust backdoor-based watermarking method for FedGL. FedGMark comprises two main components: *Customized Watermark Generator (CWG)* and *Robust Model Loader (RML)* (as depicted in Figure 1). The CWG module utilizes the unique property of each client, as different clients could have different properties (e.g., distributions of their graph data) and their optimal watermark could be different. Particularly, CWG learns the customized watermark for each graph using its structure information, and outputs a set of diversified watermarked graphs for each client. Further, inspired by existing GNNs [Xu *et al.*, 2019], the RML module designs a new GL model that consists of multiple submodels, each being any existing GL model. It also introduces a voting classifier for aggregating the prediction results from the submodels. Under this design, FedGMark can be proved to be certified robust against the *worst-case* layer-perturbation attack, once the number of perturbed layers is bounded. The model owner (e.g., participating clients in FedGL) adopts the designed GL model to train the local watermarked model with the learnt watermarked graphs and the remaining clean graphs. After the server-client training terminates, the ownership of the trained FedGL model can be verified via measuring its performance on a set of testing graphs injected with the *global watermark*, which is the integration of all clients' local watermarks.

3.2 Customized Watermark Generator (CWG)

CWG consists of two networks: $\mathrm{GatingNet}$ and KeyNet. $\mathrm{GatingNet}$ designs the watermark for each graph separately using the edge information, while KeyNet learns client-wise watermarking style using predefined keys (e.g., client ID in this paper). The customized watermark for each client's graph is then decided using the output of $\mathrm{GatingNet}$ and KeyNet. *Detailed network architectures of CWG can be seen in Table 7 in Appendix C.* We demonstrate how CWG can learn a customized watermark using a graph $G^i = (\mathbb{V}^i, \mathbb{E}^i)$ from client i as an instance. The details are as follows:

- We first randomly select n_w nodes $\mathbb{V}_w^i = \{v_1, \cdots, v_{n_w}\}$ from $\mathbb{V}^i$ as watermark nodes and construct a corresponding mask matrix $\mathbf{M}^i \in \{0,1\}^{|\mathbb{V}^i| \times |\mathbb{V}^i|}$ such that $\mathbf{M}^i[v_j, v_k] = 1$ if $v_j, v_k \in \mathbb{V}_w^i$, and 0 otherwise. We also update the adjacency matrix $\mathbf{A}^i$ of G^i according to $\mathbb{V}_w^i$, i.e., setting $\mathbf{A}^i[v_j, v_k] = 0, \forall v_j, v_k \in \mathbb{V}_w^i$. This allows us focus on learning the edge status between watermarked nodes.

- Given the client i's ID string k^i, we utilize a cryptographic hash function, such as MD5, to convert it into an integer (e.g., 128-bit long with the integer range $[0, 2^{128} - 1]$). This integer is then employed as a seed to produce a key matrix $\mathbf{K}^i \in \mathbb{R}^{|\mathbb{V}^i| \times |\mathbb{V}^i|}$. Then, we employ $\mathrm{GatingNet}$ and KeyNet to extract edge features and key features, resulting in $\tilde{\mathbf{A}}^i = \mathrm{GatingNet}(\mathbf{A}^i) \in [0,1]^{|\mathbb{V}^i| \times |\mathbb{V}^i|}$ and $\tilde{\mathbf{K}}^i = \mathrm{KeyNet}(\mathbf{K}^i) \in [0,1]^{|\mathbb{V}^i| \times |\mathbb{V}^i|}$, respectively.

- We finally learn the customized watermark for G^i by integrating $\tilde{\mathbf{A}}^i$, $\tilde{\mathbf{K}}^i$, and $\mathbf{M}^i$, and obtain the corresponding watermarked graph as $G_w^i = (\mathbb{V}^i, \mathbb{E}_w^i)$. Here $\mathbb{E}_w^i$ is the set of edges according to the updated adjacency matrix $\mathbf{A}^i \oplus \mathbf{W}^i$, where $\oplus$ is the element-wise addition and $\mathbf{W}^i = \mathbb{I}((\tilde{\mathbf{A}}^i \odot \tilde{\mathbf{K}}^i) > 0.5) \odot \mathbf{M}^i$ contains the edge status between the watermark nodes $\mathbb{V}_w^i$. Here, $\odot$ is the element-wise product, $\mathbb{I}(p)$ is an indicator function returning 1 if p is true, and 0 otherwise. We adopt 0.5 as a threshold to decide the presence of edges between watermarked nodes.

3.3 Robust Model Loader (RML)

This module aims to design a new GL model that is provably robust to the layer-perturbation attack. Towards this end, we design a GL model architecture to incorporate multiple submodels; and devise a majority voting-based ensemble classifier on top of the predictions of these submodels.

Architecture of the proposed GL model. Intuitively, each client can take a base GL model (e.g., GIN [Xu *et al.*, 2019]) and split it according to the layer indexes to obtain multiple submodels. For instance, a 8-layer GIN can be represented with layer indexes $\{l_1, \cdots, l_8\}$. Splitting this GIN into 4 submodels $\{\mathrm{GIN}_1, \cdots, \mathrm{GIN}_4\}$ with layer indexes $\{l_1, l_2\}, \cdots, \{l_7, l_8\}$ means GIN_i contains layers

$\{l_{2i-1}, l_{2i}\}$, from the GIN. However, submodels splitted in this way are coupled from each other, making them unable to defend against layer-perturbation attacks. To tackle this problem, we design the novel GL model θ that is an ensemble of a set of S *independent* submodels $\{\vartheta_1, \vartheta_2, \cdots, \vartheta_S\}$, where each submodel ϑ_i is a base GL model. This approach can hence be easily adapted to any existing FedGL. Further, to prevent homogeneity, we define varying channels for these submodels to diversify them. Details of the model architecture are shown in Table 8 in Appendix C.

A majority-voting based ensemble classifier. The designed GL model architecture inspires us to leverage the idea of ensemble classifier, which can combine the predictions of base "weak" classifiers. Specifically, we propose a majority voting-based ensemble classifier to combine the predictions of the submodels. Given a testing graph G and a graph classifier f, we denote the prediction of the submodel ϑ_i for G as $y = f(\vartheta_i, G) \in \mathbb{Y}$. For a GL model θ with S submodels $\{\vartheta_1, \vartheta_2, \cdots, \vartheta_S\}$, we can count the submodels that classify G to be y as $N_y = \sum_{i=1}^{S} \mathbb{I}(f(\vartheta_i, G) = y)$. Then we introduce our majority-voting based ensemble classifier g to classify G as: $g(\theta, G) = \arg\max_{y \in \mathbb{Y}} N_y$. In cases of ties, our ensemble classifier g selects the label with a smaller index.

3.4 Training the Proposed FedGL Model

The overall training process consists of three iterative steps: 1) training the proposed GL model in all clients; 2) training the CWG module in *watermarked clients*, i.e., the clients that aim to inject watermarked graphs for protecting the model ownership; and 3) aggregating the clients' GL models to produce the target watermarked model. The final global model is the learnt watermarked FedGL model. Details of training can be seen in Algorithm 1 in the Appendix.

Step 1: Training the proposed GL model. Assume we have T_w watermarked clients with indexes $[1, T_w]$. For each watermarked client i, we split its training graphs $\mathbb{G}^i$ into the watermarked graphs $\mathbb{G}^i_w$ with a target label, say y_w, and remaining clean graphs $\mathbb{G}^i_c$, and then customize the watermark for each graph in $\mathbb{G}^i_w$ using the CWG module (see **Step 2**). Given the client's GL model θ^i with S submodels $\{\vartheta_1^i, \vartheta_2^i, \cdots, \vartheta_S^i\}$, we train each submodel ϑ_j^i via minimizing the loss on $\mathbb{G}^i_c$ and $\mathbb{G}^i_w$, i.e., $\vartheta_j^i = \arg\min_{\vartheta_j^i} L(\vartheta_j^i; \mathbb{G}^i_c \bigcup \mathbb{G}^i_w)$. For an unwatermarked client $k \in [T_w + 1, T]$, we utilize all clean graphs $\mathbb{G}^k$ to train each submodel ϑ_j^k separately, i.e., $\vartheta_j^k = \arg\min_{\vartheta_j^k} L(\vartheta_j^k; \mathbb{G}^k)$.

Step 2: Training the CWG. We denote the parameters of the CWG module for a watermarked client $i \in [1, T_w]$, as ω^i. The parameters include two networks, $\mathrm{GatingNet}$ and KeyNet. Each client i trains its CWG ω^i to ensure that the generated watermarks be effective and diverse. Formally, we have $\omega^i = \arg\min_{\omega^i} L(\theta^i; \mathbb{G}^i_w), i \in [1, T_w]$.

Step 3: Aggregating clients' GL models. The server averages GL models $\{\theta^i\}_{i \in T}$ to produce the global model θ, and distributes this model to selected clients in the next iteration.

3.5 Model Ownership Verification

When suspecting the target FedGMark model θ is illegally used by others, the model owner (all the participating clients or their representative) can recruit a trusted judge for model ownership verification. Typically, the judge requests both the true model owner and the illegal party to provide some test data for verification. Only when the one knows the predictions by the target model for the provided test data by both parties, the judge will confirm this party the model ownership. In particular, besides providing the clean data $\mathbb{G}^i_c$ by both parties that behave normally, the true model owner especially provides the designed watermarked data $\mathbb{G}^i_w$ that only s/he knows the model behaves on. As a result, both parties know the prediction results on $\mathbb{G}^i_c$, but the illegal party is hard to predict accurately on $\mathbb{G}^i_w$ provided by the true model owner.

3.6 Certified Robustness Guarantees against Layer-Perturbation Attacks

We show the above design, with any layer-perturbation attack, ensures the predictions of the learnt watermarked FedGL model and its compromised counterpart for the watermarked graphs are consistent, once the number of perturbed layers is bounded. Given the target watermarked FedGL model θ and its S submodels $\{\vartheta_1, \cdots, \vartheta_S\}$, we denote θ' as the comprised model and $\{\vartheta_1', \cdots, \vartheta_S'\}$ as its S submodels. For each watermarked graph G_w, we use the ensemble classifier g on submodels' predictions, i.e., its predictions on θ and θ' are $g(\theta, G_w) = \arg\max_{y \in \mathbb{Y}} N_y$, and $g(\theta', G_w) = \arg\max_{y \in \mathbb{Y}} N_y'$, respectively, where $N_y = \sum_{i=1}^{S} \mathbb{I}(f(\vartheta_i, G) = y)$ and $N_y' = \sum_{i=1}^{S} \mathbb{I}(f(\vartheta_i', G) = y)$. Then we have the following result on guaranteeing the number of perturbed layers on the target watermarked model.

Theorem 1 (Certified number of perturbed layers r.) *Let θ, θ', g, and G_w be above defined. Suppose N_A and N_B are the largest and second largest count outputted by g on G_w, For any layer-perturbation attack, we have $g(\theta, G_w) = g(\theta', G_w)$, when the number of perturbed layers r satisfies:*

$$r \leq r^* = \left(N_A - N_B + \mathbb{I}[A < B] - 1\right)/2, \tag{1}$$

where $\mathbb{I}[\cdot]$ is the indicator function and r is called the certified number of perturbed layers.

We also show the tightness of our derived r^* in the following theorem:

Theorem 2 (Tightness of r^*.) *Without using extra information of f, our derived r^* in Theorem 1 is tight. I.e., r^* is the maximum number of perturbed layers tolerated by our target watermarked model.*

The proofs of Theorems 1 and 2 are deferred to Appendix A.

4 Experiments

In this section, we comprehensively evaluate FedGMark on multiple datasets, FedGL models, attack baselines, and experimental settings. More experimental results and discussions are in Appendix.

4.1 Experimental Setup

Datasets and models. We evaluate our FedGMark on four real-world graph datasets for graph classification: MUTAG [Debnath *et al.*, 1991], PROTEINS [Borgwardt *et al.*, 2005], DD [Dobson and Doig, 2003], and COLLAB [Yanardag and Vishwanathan, 2015]. Details about the statistics of those datasets are shown in Table 6 in Appendix C. Following prior work Shen *et al.* [2022]; Xia *et al.* [2024], we choose the well-known GIN [Xu *et al.*, 2019], GSAGE [Hamilton *et al.*, 2017], and GCN [Kipf and Welling, 2017] as the GL model. All these network architectures involved in the experiments are detailed in Table 8 in Appendix C.

Parameter setting. We implement our method using one NVIDIA GeForce GTX 1080 Ti GPU. In FedGL, we use $T = 40$ clients in total and train the model 200 iterations. The server randomly selects 50% clients in each iteration. We define the target label of watermarking graphs as 1, and each participating client randomly selects 10% graphs with labels not 1 as the watermarking graphs. We extensively validate the effectiveness and robustness of the proposed watermarking method with the following hyperparameters details: the number of submodels $S = \{4, 8, 16\}$, the number watermarked clients $T_w = \{5, 10, 20\}$ (both S and T_w are halved on MUTAG due to less data), the watermarked nodes $n_w = \{3, 4, 5\}$, and the number of perturbed layers $r = \{1, \cdots, 5\}$ in the layer-perturbation attack. By default, we set $S = 4$, $T_w = 10$, $n_w = 4$, $r = 1$. While studying the impact of a hyperparameter, we fix the others as the default value.

Evaluation metric. We use three metrics for evaluation: the main task accuracy (MA), watermark accuracy (WA), and certified WA (CWA@r). An effective and robust watermarked model is expected to achieve both high MA and WA. CWA evaluates the certified robustness of FedGMark against layer-perturbation attacks. CWA@r is defined as the fraction of testing graphs that are provably predicted as the target label, when at most r layers in client models can be arbitrarily perturbed.

Attack baselines. We evaluate FedGMark against existing watermark removal attacks including distillation and finetuning, and our proposed layer-perturbation attack. In our setting, the layer-perturbation attack can replace *any* layer(s)' parameters of the target watermarked model with those from the unwatermarked model to maximally reduce the watermark accuracy on its watermarked graphs. When perturbing multiple layers, we utilize a greedy algorithm to decide the perturbed layers—we search for one optimal perturbed layer at each step. Specifically, we first traverse perturbing layers in the watermarked model and find the one that maximally reduces the watermark accuracy. We then search for the remaining layers based on this one and continue the process.

4.2 Empirical Results: MA and WA

4.2.1 Results under the Default Setting

We first assess our FedGMark against empirical and layer-perturbation attacks under the default setting. Experimental results are provided in Table 2. We have the following observations:

- **1) Our FedGMark significantly outperforms the existing method under no attack.** Recall in Table 1 that [Xu *et al.*, 2023] obtains $< 60\%$ WAs across all datasets and FedGL models. In contrast, FedGMark can achieve WAs $> 70\%$ in almost all cases, while having similar MAs.

Table 2: Results of our FedGMark under empirical watermark removal attacks.

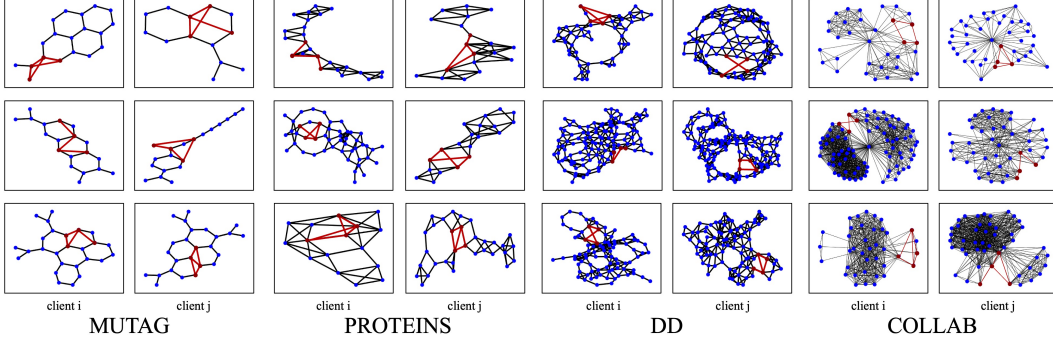| Datasets	Attack	Fed-GIN		Fed-GSAGE		Fed-GCN		Datasets	Attack	Fed-GIN		Fed-GSAGE		Fed-GCN	
		MA↑	WA↑	MA↑	WA↑	MA↑	WA↑			MA↑	WA↑	MA↑	WA↑	MA↑	WA↑
MUTAG	None	**0.81**	**0.90**	**0.85**	**0.89**	**0.83**	**0.83**	PROTEINS	None	**0.72**	**0.86**	**0.72**	**0.82**	**0.73**	**0.84**
	Distillation	0.83	0.87	0.81	0.70	0.80	0.80		Distillation	0.70	0.84	0.70	0.80	0.70	0.79
	Finetuning	0.82	0.88	0.79	0.76	0.78	0.81		Finetuning	0.70	0.84	0.71	0.82	0.72	0.80
	1-Layer Pert.	**0.82**	**0.90**	**0.83**	**0.91**	**0.81**	**0.84**		1-Layer Pert.	**0.73**	**0.86**	**0.74**	**0.83**	**0.74**	**0.85**
DD	None	**0.73**	**0.65**	**0.74**	**0.57**	**0.74**	**0.56**	COLLAB	None	**0.73**	**0.75**	**0.74**	**0.74**	**0.73**	**0.70**
	Distillation	0.71	0.63	0.74	0.56	0.72	0.56		Distillation	0.73	0.71	0.72	0.73	0.70	0.69
	Finetuning	0.72	0.62	0.74	0.57	0.74	0.55		Finetuning	0.72	0.70	0.75	0.71	0.72	0.67
	1-Layer Pert.	**0.72**	**0.65**	**0.75**	**0.55**	**0.73**	**0.57**		1-Layer Pert.	**0.72**	**0.76**	**0.73**	**0.74**	**0.74**	**0.71**

Figure 2: Example learnt watermarks and watermarked graphs by our FedGMark. CWGs generated by different clients produce unique watermarks, characterized by distinct edge connection patterns.

Table 3: Impact of S on FedGMark against our layer-perturbation attack.

Datasets ($S = 8$)	Attack	Fed-GIN		Fed-GSAGE		Fed-GCN		Datasets	Attack	Fed-GIN		Fed-GSAGE		Fed-GCN	
		MA↑	WA↑	MA↑	WA↑	MA↑	WA↑			MA↑	WA↑	MA↑	WA↑	MA↑	WA↑
MUTAG	None	**0.83**	**0.93**	**0.83**	**0.92**	**0.78**	**0.87**	PROTEINS	None	**0.73**	**0.86**	**0.71**	**0.85**	**0.74**	**0.88**
	1-Layer Pert.	0.79	0.93	0.84	0.92	0.79	0.76		1-Layer Pert.	0.70	0.86	0.71	0.85	0.73	0.88
DD	None	**0.71**	**0.57**	**0.73**	**0.52**	**0.74**	**0.50**	COLLAB	None	**0.73**	**0.73**	**0.74**	**0.74**	**0.72**	**0.72**
	1-Layer Pert.	0.72	0.56	0.73	0.50	0.74	0.50		1-Layer Pert.	0.73	0.72	0.74	0.75	0.72	0.72
($S = 16$)	None	**0.83**	**0.90**	**0.86**	**0.87**	**0.79**	**0.79**	PROTEINS	None	**0.72**	**0.86**	**0.72**	**0.85**	**0.71**	**0.85**
MUTAG	1-Layer Pert.	0.81	0.90	0.84	0.86	0.79	0.80		1-Layer Pert.	0.73	0.86	0.72	0.85	0.73	0.85
DD	None	**0.70**	**0.61**	**0.74**	**0.54**	**0.73**	**0.50**	COLLAB	None	**0.71**	**0.76**	**0.72**	**0.74**	**0.72**	**0.72**
	1-Layer Pert.	0.73	0.61	0.74	0.54	0.74	0.50		1-Layer Pert.	0.71	0.76	0.71	0.73	0.72	0.72

Such significant improvements confirm the superiority of our learnt watermarks over the random watermarks in [Xu *et al.*, 2023]. Figure 2 also visualizes example learnt watermarks and we can see these watermarks are diversified due to the proposed CWG.

- **2) Our FedGMark exhibits resistance to existing empirical attacks.** As shown in Table 1, existing methods are vulnerable to distillation and finetuning attacks. In contrast, the WAs of FedGMark under these two attacks are almost the same as those under no attack, demonstrating FedGMark is robust to the existing attacks.

- **3) Our FedGMark is resilient to the proposed layer perturbation attack.** We notice that the existing method has difficulties in defending against the proposed layer-perturbation attacks and shows an unsatisfactory WA, e.g., $< 25\%$ WA in almost all cases under the 1-layer perturbation attack. Conversely, our FedGMark can obtain a close WA compared to that without attack. This verifies our ensemble classifier in FedGMark is capable of resisting the 1-layer perturbation attack.

4.2.2 Impact of Hyperparameters

This section studies the impact of hyperparameters in FedGMark against the layer-perturbation attack.

Impact of #submodels S. We first examine the impact of S on the performance of FedGMark and report the results against the 1-layer perturbation attack in Table 3. We observe that the learnt watermarked model can resist to the 1-layer perturbation attack under all S and both the MAs and WAs are also similar. This indicates the number of submodels marginally affects the watermarking performance against the 1-layer perturbation attack.

Table 4: Impact of #perturbed layers on FedGMark against our layer-perturbation attack. Compared with Table 2, the change in MA is less than 4% in all cases.

| Datasets	#Perturbed Layers (Net)	1 WA↑	2 WA↑	3 WA↑	4 WA↑	5 WA↑	Datasets	#Perturbed Layers (Net)	1 WA↑	2 WA↑	3 WA↑	4 WA↑	5 WA↑
MUTAG	Fed-GIN	0.90	0.89	0.54	0.23	0.16	PROTEINS	Fed-GIN	0.86	0.84	0.58	0.46	0.39
	Fed-GSAGE	0.91	0.89	0.55	0.21	0.13		Fed-GSAGE	0.83	0.81	0.55	0.45	0.44
	Fed-GCN	0.84	0.83	0.49	0.24	0.18		Fed-GCN	0.85	0.82	0.57	0.48	0.41
DD	Fed-GIN	0.65	0.61	0.56	0.32	0.21	COLLAB	Fed-GIN	0.76	0.74	0.58	0.36	0.27
	Fed-GSAGE	0.55	0.55	0.49	0.28	0.24		Fed-GSAGE	0.74	0.73	0.57	0.35	0.25
	Fed-GCN	0.57	0.54	0.48	0.29	0.24		Fed-GCN	0.71	0.70	0.52	0.32	0.21
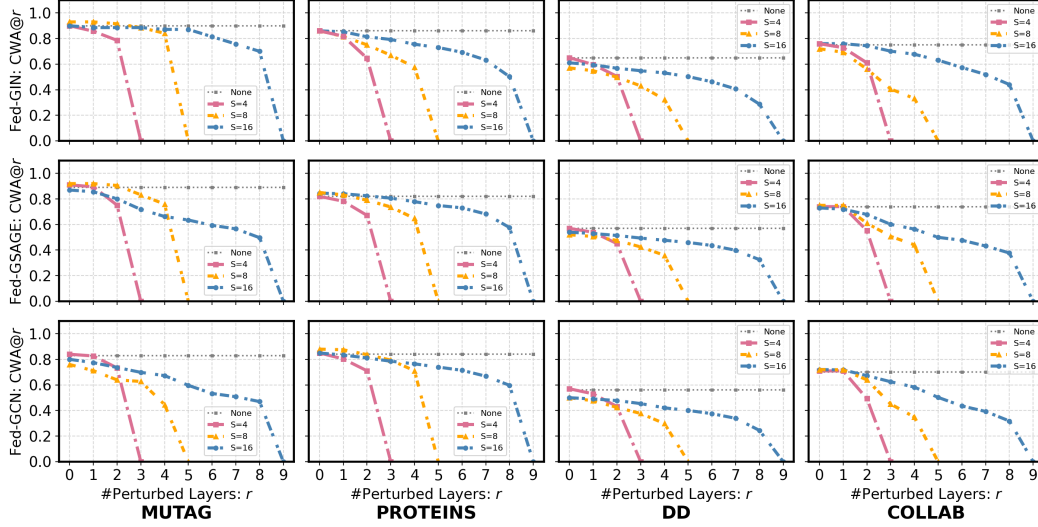
Figure 3: CWA vs. #perturbed layers r in the layer-perturbation attack.

Impact of #watermarking clients T_w. The results on $T_w = 5$ and $T_w = 20$ are shown in Figure 4 in Appendix. We see that: *the larger number of clients generating watermarks, the better our watermarking performance.* For instance, the WA improves by 11%, 9%, and 7%, respectively on the three FedGL models on PROTEINS when T_w increases from 5 to 20. This is because more watermark clients (thus more watermarked graphs) can ensure the FedGL model better learns the relation between the watermarks and the target label.

Impact of the watermark size n_w. We also investigate $n_w = 3$ and $n_w = 5$, and the results are presented in Figure 5 in Appendix. Similarly, FedGMark obtains higher WA with larger n_w. This is because a larger watermark size can facilitate the trained FedGL model better learn the watermark.

Impact of #perturbed layers. Finally, Table 4 shows the results of FedGMark vs. different #perturbed layers. We observe that: 1) The WA decreases as increasing the #perturbed layers. This is because the attacker has more attack capability by perturbing more layers. 2) Despite the theoretically guaranteed CWA being 0 at #perturbed layers ≥ 3 as later depicted in the Figure 3, our empirical watermarking performance can still achieves WA from 48% to 58%. Note that the change in MA is $< 4\%$ in all cases, compared with Table 2.

4.2.3 Ablation Study

We investigate the contribution of each module in FedGMark, including the GatingNet and KeyNet in CWG, and the submodels used in RML. The results under the default setting and no attack are shown in Table 5. First, GatingNet plays a crucial role on generating more

Table 5: Impact of different modules in FedGMark.

Models	G.Net	K.Net	RML	MUTAG MA↑	MUTAG WA↑	PROTEINS MA↑	PROTEINS WA↑	DD MA↑	DD WA↑	COLLAB MA↑	COLLAB WA↑
(a)	✓	✓	✓	**0.81**	**0.90**	**0.72**	**0.86**	**0.73**	**0.65**	**0.73**	**0.75**
(b)	✓		✓	0.81	0.84	0.72	0.80	0.72	0.57	0.72	0.70
(c)		✓	✓	0.80	0.69	0.72	0.68	0.72	0.48	0.73	0.67
(d)	✓	✓		0.81	0.89	0.73	0.86	0.71	0.66	0.72	0.75
(e)	✓			0.82	0.47	0.71	0.56	0.73	0.39	0.73	0.57

effective watermarks and results in an improvement in WA from 8% to 21%. Second, KeyNet facilitates watermark differentiation between clients, leading to an improve on WA from 5% to 8%. Third, the submodels in RML has a negligible impact on WA/MA. However, it is important to defend against the layer-perturbation attack, as shown in Section 4.3.

4.3 Certified Robustness Results: Certified WA

In this section, we evaluate the certified robustness of FedGMark against the layer perturbation attack. The CWAs on the three FedGL models and four datasets are depicted in Figure 3. We have several observations: 1) FedGMark achieves promising provable robustness results against the *worst-case* layer perturbation attack, when the #perturbed layers is within the certified range in Eqn (1). For instance, when $S = 4$ and $0 < r \leq 2$, CWA is close to WA without attack ($r = 0$). 2) As S increases, the certified #perturbed layers also increases, showing that a more number of submodules in RML can better provably defend against the layer-perturbation attack. For instance, when $S = 16$, the CWA is 86% on MUTAG even when 5 any layers in the global model are arbitrarily perturbed, while it is 0 when $S = 4$. However, this is at the cost of requiring more computational resources (e.g., GPU memory, runtime). As shown in Table 10 in Appendix, the runtime is linearly to S.

5 Related Work

Backdoor-based watermarking for centralized models on non-graph data. Many backdoor-based watermarking methods [Lv *et al.*, 2023; Yan *et al.*, 2023; Tekgul *et al.*, 2021; Yang *et al.*, 2023; Li *et al.*, 2022a; Shao *et al.*, 2022; Lansari *et al.*, 2023] have been proposed that can *empirically* protect the model ownership. These methods mainly focus on centralized learning models on non-graph (e.g., image) data. Compared with non-graph data, graph data have unique graph structure information, e.g., entities are connected by links. Similarly, compared with centralized models, FL models are collaboratively trained by multiple clients, which could have their own uniqueness. Bansal *et al.* [2022] is the first watermarking method for centralized non-graph models with certified guarantee. However, its certified robustness performance is unsatisfactory.

Backdoor-based watermarking for FL models on non-graph data. A few recent works Tekgul *et al.* [2021]; Li *et al.* [2022a]; Bansal *et al.* [2022] design backdoor-based watermarks for protecting the ownership of FL models on non-graph data, where the watermark can be injected into client's data or server's validation data. For instance, Tekgul *et al.* [2021] presented WAFFLE, an approach to watermark DNN models by incorporating a re-training step via watermarked data at the server. Li *et al.* [2022a] leveraged each client's private watermark to verify FL model ownership, ensuring non-conflicting watermarks across different clients. However, these techniques cannot be directly applied to graph data, as they often require fixed input data, while graph data often have varying sizes. Moreover, all these methods do not provide guaranteed watermarking performance under the attack.

Backdoor-based watermarking for centralized GL models on graph data. The only work [Xu *et al.*, 2023] handling graph data uses random graph as the watermark. This method can be extended to the FedGL model, but its watermarking performance is far from satisfactory–especially vulnerable to watermark removal attacks such as distillation, finetuning, and the layer perturbation attack.

Backdoor attacks for centralized and federated GL models on graph data. Several works [Zhang *et al.*, 2021; Yang *et al.*, 2024] design backdoor attacks to manipulate GL models, enabling the attacker to influence the learned model to serve its purpose–the model will predict the attacker-chosen label for test graphs once they contain a predefined trigger. For instance, Zhang *et al.* [2021] uses random subgraphs as the trigger to backdoor centralized GL models, while Yang *et al.* [2024] learns subgraph trigger to backdoor FedGL models. Note that the goal of these works is orthogonal to ours.

6 Conclusion

We protect the model ownership of emerging FedGL trained on distributed graph data, and use the *de facto* backdoor-based watermarking method. We develop the first certifiably robust backdoor-based watermarking method FedGMark for FedGL. FedGMark demonstrates the capability of achieving high empirical watermarking performance under no attack, under existing backdoor removal attacks and the proposed stronger layer-perturbation attack. FedGMark is also provably robust against the worst-case layer-perturbation attack, once the number of perturbed layers is bounded by Theorem 1.

References

Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1615–1631, 2018.

Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. Personalized subgraph federated learning. In *International Conference on Machine Learning*, pages 1396–1415. PMLR, 2023.

Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International conference on artificial intelligence and statistics*, pages 2938–2948. PMLR, 2020.

Arpit Bansal, Ping-yeh Chiang, Michael J Curry, Rajiv Jain, Curtis Wigington, Varun Manjunatha, John P Dickerson, and Tom Goldstein. Certified neural network watermarks with randomized smoothing. In *International Conference on Machine Learning*, pages 1450–1465. PMLR, 2022.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.

Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

Edgar N Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Chaoyang He, Emir Ceyani, Keshav Balasubramanian, Murali Annavaram, and Salman Avestimehr. Spreadgnn: Decentralized multi-task federated learning for graph neural networks on molecular data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6865–6873, 2022.

Zhengyuan Jiang, Minghong Fang, and Neil Zhenqiang Gong. Ipcert: Provably robust intellectual property protection for machine learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3612–3621, 2023.

Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *In International Conference on Learning Representations (ICLR)*, 2017.

Mohammed Lansari, Reda Bellafqira, Katarzyna Kapusta, Vincent Thouvenot, Olivier Bettan, and Gouenou Coatrieux. When federated learning meets watermarking: A comprehensive overview of techniques for intellectual property protection. *Machine Learning and Knowledge Extraction*, 5(4):1382–1406, 2023.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. Ditto: Fair and robust federated learning through personalization. In *International conference on machine learning*, pages 6357–6368. PMLR, 2021.

Bowen Li, Lixin Fan, Hanlin Gu, Jie Li, and Qiang Yang. Fedipr: Ownership verification for federated deep neural network models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4521–4536, 2022.

Zheng Li, Muhammad Bilal, Xiaolong Xu, Jielin Jiang, and Yan Cui. Federated learning-based cross-enterprise recommendation with graph neural networks. *IEEE Transactions on Industrial Informatics*, 19(1):673–682, 2022.

Peizhuo Lv, Pan Li, Shengzhi Zhang, Kai Chen, Ruigang Liang, Hualong Ma, Yue Zhao, and Yingjiu Li. A robustness-assured white-box watermark in neural networks. *IEEE Transactions on Dependable and Secure Computing*, 2023.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

Liang Peng, Nan Wang, Nicha Dvornek, Xiaofeng Zhu, and Xiaoxiao Li. Fedni: Federated graph learning with network inpainting for population-based disease prediction. *IEEE Transactions on Medical Imaging*, 2022.

Aniruddha Saha, Akshayvarun Subramanya, and Hamed Pirsiavash. Hidden trigger backdoor attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11957–11965, 2020.

Masoumeh Shafieinejad, Nils Lukas, Jiaqi Wang, Xinda Li, and Florian Kerschbaum. On the robustness of backdoor-based watermarking in deep neural networks. In *Proceedings of the 2021 ACM workshop on information hiding and multimedia security*, pages 177–188, 2021.

Shuo Shao, Wenyuan Yang, Hanlin Gu, Zhan Qin, Lixin Fan, Qiang Yang, and Kui Ren. Fedtracker: Furnishing ownership verification and traceability for federated learning model. *arXiv preprint arXiv:2211.07160*, 2022.

Yun Shen, Xinlei He, Yufei Han, and Yang Zhang. Model stealing attacks against inductive graph neural networks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1175–1192. IEEE, 2022.

Yue Tan, Yixin Liu, Guodong Long, Jing Jiang, Qinghua Lu, and Chengqi Zhang. Federated learning on non-iid graphs via structural knowledge sharing. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 9953–9961, 2023.

Buse GA Tekgul, Yuxi Xia, Samuel Marchal, and N Asokan. Waffle: Watermarking in federated learning. In *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*, pages 310–320. IEEE, 2021.

Yusuke Uchida, Yuki Nagai, Shigeyuki Sakazawa, and Shin'ichi Satoh. Embedding watermarks into deep neural networks. In *Proceedings of the 2017 ACM on international conference on multimedia retrieval*, pages 269–277, 2017.

Vidya et al. Fedml supports several out-of-the-box deep learning algorithms for various data types, such as tabular, text, image, graphs, and internet of things (iot) data. `https://aws.amazon.com/blogs/machine-learning/part-2-federated-learning-on-aws-with-fedml-health-analytics-without-sharing-sensitive-data/`.

Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in Neural Information Processing Systems*, 33:16070–16084, 2020.

Binghui Wang, Ang Li, Meng Pang, Hai Li, and Yiran Chen. Graphfl: A federated learning framework for semi-supervised node classification on graphs. In *2022 IEEE International Conference on Data Mining (ICDM)*, pages 498–507. IEEE, 2022.

Zhen Wang, Weirui Kuang, Yuexiang Xie, Liuyi Yao, Yaliang Li, Bolin Ding, and Jingren Zhou. Federatedscope-gnn: Towards a unified, comprehensive and efficient package for federated graph learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4110–4120, 2022.

Peter Wills and François G Meyer. Metrics for graph comparison: a practitioner's guide. *Plos one*, 15(2):e0228728, 2020.

Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Tao Qi, Yongfeng Huang, and Xing Xie. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications*, 13(1):3091, 2022.

Zaishuo Xia, Han Yang, Binghui Wang, and Jinyuan Jia. Gnncert: Deterministic certification of graph neural networks against adversarial perturbations. In *International Conference on Learning Representations*, 2024.

Han Xie, Jing Ma, Li Xiong, and Carl Yang. Federated graph classification over non-iid graphs. *Advances in neural information processing systems*, 34:18839–18852, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Jing Xu, Stefanos Koffas, Oğuzhan Ersoy, and Stjepan Picek. Watermarking graph neural networks based on backdoor attacks. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, pages 1179–1197. IEEE, 2023.

Yifan Yan, Xudong Pan, Mi Zhang, and Min Yang. Rethinking {White-Box} watermarks on deep learning models under neural structural obfuscation. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 2347–2364, 2023.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.

Wenyuan Yang, Shuo Shao, Yue Yang, Xiyao Liu, Ximeng Liu, Zhihua Xia, Gerald Schaefer, and Hui Fang. Watermarking in secure federated learning: A verification framework based on client-side backdooring. *ACM Transactions on Intelligent Systems and Technology*, 15(1):1–25, 2023.

Yuxin Yang, Qiang Li, Jinyuan Jia, Yuan Hong, and Binghui Wang. Distributed backdoor attacks on federated graph learning and certified defenses. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security*, 2024.

Yuhang Yao, Weizhao Jin, Srivatsan Ravi, and Carlee Joe-Wong. Fedgcn: Convergence-communication tradeoffs in federated training of graph convolutional networks. *Advances in Neural Information Processing Systems*, 36, 2024.

Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International conference on machine learning*, pages 5650–5659. Pmlr, 2018.

Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Backdoor attacks to graph neural networks. In *SACMAT*, 2021.

A Proofs

Proof of Theorem 1: Given a watermarked graph G_w, a base graph classifier f, our ensemble classifier g, and the target watermarked FedGL model $\theta = \{\vartheta_i\}_{i=1}^S$. Let θ' be the watermarked FedGL model after the layer-perturbation attack. We denote by $y = f(\vartheta_i, G_w)$ the prediction of G_w by submodel i, and $g(\theta, G_w) = \arg\max_{y \in \mathbb{Y}} N_y$ the prediction of G_w by the target model θ, where $N_y = \sum_{i=1}^S \mathbb{I}(f(\vartheta_i, G_w) = y)$ and $\theta = \{\vartheta_i\}_{i=1}^S$. Assume N_A and N_B are the largest and second largest votes of the output of the ensemble classifier g on the target model θ and G_w, respectively. Similarly, let N_A' and N_B' respectively denote the corresponding votes of the model θ'. Under our GL model architecture, each perturbed layer affects at most 1 submodel in the worst case. Therefore, N_A' and N_B' satisfy the following equations when r layers in θ are perturbed:

$$
\begin{aligned}
N_A - r \le N_A' \le N_A + r, \\
N_B - r \le N_B' \le N_B + r.
\end{aligned}
\tag{2}
$$

Thus, $g(\theta', G) = g(\theta, G)$ when $N_A' > N_B'$, This means $N_A - r > N_B + r - \mathbb{1}(A < B)$, where $\mathbb{I}(A < B)$ indicates we select a label with a smaller index in case of ties. Hence, r satisfies:

$$
r \le r^* = \frac{N_A - N_B + \mathbb{1}(A < B) - 1}{2}.
\tag{3}
$$

Proof of Theorem 2: We use proof by contradiction, i.e., when $r > r^*$, we can build a base classifier f' such that there exists $g(\theta, G_w) \ne g(\theta', G_w)$. We select the smallest $r = r^* + 1$ for simplicity, indicating that $r^* + 1$ layers whose model parameters can be arbitrarily perturbed. Assume there exist a graph classifier f' under which, $(r^* + 1)$ submodels whose original predictions on G_w are the label A and now become B after $r^* + 1$ layers are perturbed. Then we have $N_A' = N_A - (r^* + 1)$ and $N_B' = N_B + (r^* + 1)$. Since r^* is the largest number such that Equation (2) is satisfied, i.e., $N_A - r^* > N_B - \mathbb{1}(A < B) + r^*$ holds. Thus,

$$
N_A - (r^* + 1) \le N_B - \mathbb{1}(A < B) + (r^* + 1).
\tag{4}
$$

Or $N_A' \le N_B'$. Hence, $g(\theta, G_w) = A \ne B = g(\theta', G_w)$, if $A > B$.

B Algorithm 1

Algorithm 1 The training process of FedGMark

Input: clients $[1, T]$, watermarked clients $[1, T_w]$, training graphs $\mathbb{G}^i$ ($i \in [1, T]$), initial global model θ_1, initial CWG ω_1, #Perturbed Layers r, unique key k^i for client i ($i \in [1, T_w]$), the number of submodels S.
Output: Target global model θ_{epoch}, and target local models θ_{epoch}^j ($j \in [1, T_w]$).

1: **for** each epoch e in $[1, epoch]$ **do**
2:     **for** each client $i \in [1, T]$ **do**
3:         **if** $i \in [1, T_w]$ **then**
4:             Divide $\mathbb{G}^i$ into $\mathbb{G}_c^i$ and $\mathbb{G}_w^i$. ▷ clean & watermarked samples
5:             $\mathbb{G}_w^i = \text{CWG}(\mathbb{G}_w^i, k^i)$
6:         **end if**
7:         $\vartheta_1, \vartheta_2, \cdots, \vartheta_S = \text{divide}(\theta_e)$
8:         **for** each submodel j in $[1, S]$ **do**
9:             $\vartheta_j^i = \arg\min_{\vartheta_j^i} L(\vartheta_j; \mathbb{G}^i)$
10:         **end for**
11:         $\theta_e^i = \text{joint}(\vartheta_1^i, \vartheta_2^i, \cdots, \vartheta_S^i)$
12:         **if** $i \in [1, T_w]$ **then**
13:             $\omega_e^i = \arg\min_{\omega^i} L(\theta_e^i; \mathbb{G}_w^i)$
14:         **end if**
15:     **end for**
16:     $\theta_{e+1} = \frac{1}{|\mathbb{T}_e|} \sum_{i \in \mathbb{T}_e} \theta_e^i$ ▷ Server selects $\mathbb{T}_e$ clients for aggregation
17: **end for**

Table 6: Statistics of datasets.

Datasets	#Graphs	#Classes	Avg. #Node	Avg. # Edge	#Training graphs			#Testing graphs		
					1	2	3	1	2	3
MUTAG	188	2	17.93	19.80	83	42	-	42	21	-
PROTEINS	1110	2	37.72	70.35	440	300	-	220	150	-
DD	950	2	208.3	518.76	330	303	-	165	152	-
COLLAB	4981	3	73.49	2336.66	517	1589	1215	258	794	608

Table 7: Detailed network architectures of CWG.

Name	Network Architectures	Input	Output
GatingNet	2×(Linear-$|\mathbb{V}|$& ReLU & Dropout-0.05) Linear-$|\mathbb{V}|$ & Sigmoid	$\mathbf{A}^i \in \{0,1\}^{|\mathbb{V}|\times|\mathbb{V}|}$	$\tilde{\mathbf{A}}^i \in [0,1]^{|\mathbb{V}|\times|\mathbb{V}|}$
KeyNet	2×(Linear-$|\mathbb{V}|$& ReLU & Dropout-0.05) Linear-$|\mathbb{V}|$ & Sigmoid	$\mathbf{K}^i \in \mathbb{R}^{|\mathbb{V}|\times|\mathbb{V}|}$	$\tilde{\mathbf{K}}^i \in [0,1]^{|\mathbb{V}|\times|\mathbb{V}|}$

Table 8: Detailed network architectures for $S = \{4, 8, 16\}$ in Fed-GIN, Fed-GSAGE, and Fed-GCN models within our watermark. The convolutional (conv) layers in these three models are GINConv, GSAGEConv, and GCNConv, respectively. Here, x represents the input dimension, while y represents the output dimension. BN is short for BatchNorm.

submodels 1-4	submodels 5-8	submodels 9-16	
Linear-$x \times y$	Linear-$x \times y$	Linear-$x \times y$	Linear-$x \times y$
conv-x×64 & BN-64	conv-x×128 & BN-128	conv-x×72 & BN-72	conv-x×96 & BN-96
ReLU & Linear-64×y	ReLU & Linear-128×y	ReLU & Linear-72×y	ReLU & Linear-96×y
2×(conv-64×64 & BN-64	2×(conv-128×128 & BN-128	2×(conv-72×72 & BN-72	2×(conv-96×96 & BN-96
ReLU & Linear-64×y)	ReLU & Linear-128×y)	ReLU & Linear-72×y)	ReLU & Linear-96×y)
conv-64×64 & BN-64	conv-128×64 & BN-64	conv-72×64 & BN-64	conv-96×64 & BN-64
ReLU & Linear-64×y	ReLU & Linear-64×y	ReLU & Linear-64×y	ReLU & Linear-64×y
Linear-$x \times y$	Linear-$x \times y$	Linear-$x \times y$	Linear-$x \times y$
conv-x×64 & BN-64	conv-x×128 & BN-128	conv-x×72 & BN-72	conv-x×96 & BN-96
ReLU & Linear-64×y	ReLU & Linear-128×y	ReLU & Linear-72×y	ReLU & Linear-96×y
2×(conv-64×64 & BN-64	2×(conv-128×128 & BN-128	2×(conv-72×72 & BN-72	2×(conv-96×96 & BN-96
ReLU & Linear-64×y)	ReLU & Linear-128×y)	ReLU & Linear-72×y)	ReLU & Linear-96×y)
conv-64×32 & BN-32	conv-128×32 & BN-32	conv-72×32 & BN-32	conv-96×32 & BN-32
ReLU & Linear-32×y	ReLU & Linear-32×y	ReLU & Linear-32×y	ReLU & Linear-32×y
Linear-$x \times y$	Linear-$x \times y$	Linear-$x \times y$	Linear-$x \times y$
conv-x×64 & BN-64	conv-x×128 & BN-128	conv-x×72 & BN-72	conv-x×96 & BN-96
ReLU & Linear-64×y	ReLU & Linear-128×y	ReLU & Linear-72×y	ReLU & Linear-96×y
2×(conv-64×64 & BN-64	2×(conv-128×128 & BN-128	2×(conv-72×72 & BN-72	2×(conv-96×96 & BN-96
ReLU & Linear-64×y)	ReLU & Linear-128×y)	ReLU & Linear-72×y)	ReLU & Linear-96×y)
conv-64×16 & BN-16	conv-128×16 & BN-16	conv-72×16 & BN-16	conv-96×16 & BN-16
ReLU & Linear-16×y	ReLU & Linear-16×y	ReLU & Linear-16×y	ReLU & Linear-16×y
Linear-$x \times y$	Linear-$x \times y$	Linear-$x \times y$	Linear-$x \times y$
conv-x×64 & BN-64	conv-x×128 & BN-128	conv-x×72 & BN-72	conv-x×96 & BN-96
ReLU & Linear-64×y	ReLU & Linear-128×y	ReLU & Linear-72×y	ReLU & Linear-96×y
2×(conv-64×64 & BN-64	2×(conv-128×128 & BN-128	2×(conv-72×72 & BN-72	2×(conv-96×96 & BN-96
ReLU & Linear-64×y)	ReLU & Linear-128×y)	ReLU & Linear-72×y)	ReLU & Linear-96×y)
conv-64×8 & BN-8	conv-128×8 & BN-8	conv-72×8 & BN-8	conv-96×8 & BN-8
ReLU & Linear-8×y	ReLU & Linear-8×y	ReLU & Linear-8×y	ReLU & Linear-8×y

C   More Experimental Details

Table 6 shows the statistics of the used graph datasets. Table 7 shows the network architectures of the CWG component. Table 8 shows the detailed network architectures of Fed-GIN, Fed-GSAGE, and Fed-GCN models.
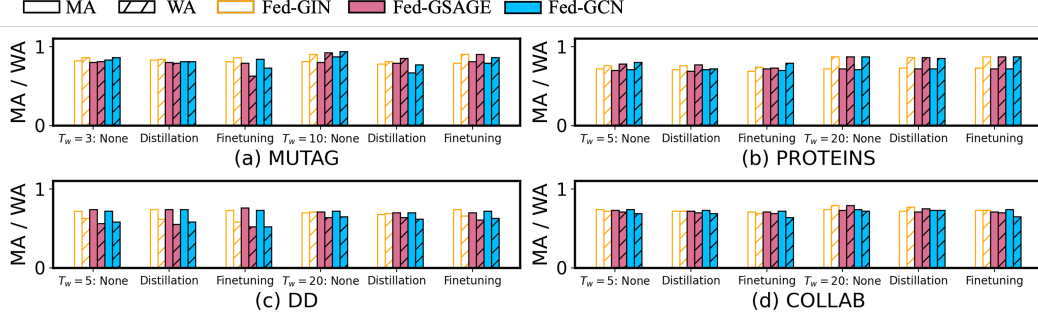
15

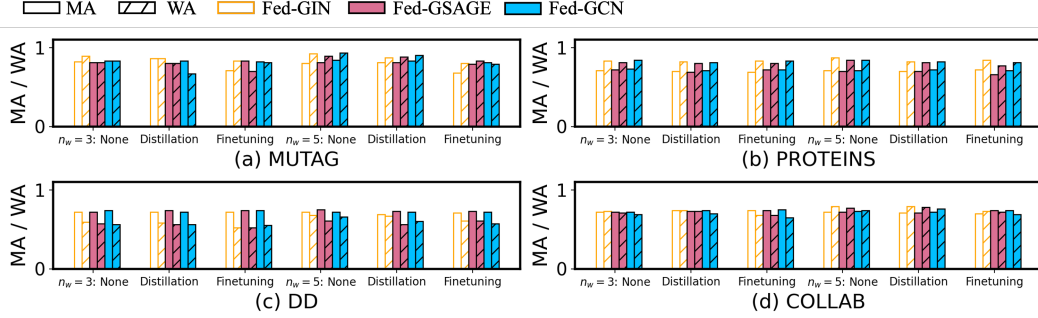Figure 4: Impact of T_w on FedGMark against prior watermark removal and layer-perturbation attacks.

Figure 5: Impact of n_w on FedGMark against prior watermark removal and layer-perturbation attacks.

Table 9: Effect of learning rate (lr) and #local epochs (le).

| Datasets | $le = 5$ | | | | | | $lr = 0.01$ | | | | | |
| | $lr = 0.01$ | | $lr = 0.05$ | | $lr = 0.1$ | | $le = 5$ | | $le = 10$ | | $le = 20$ | |
	MA↑	WA↑	MA↑	WA↑	MA↑	WA↑	MA↑	WA↑	MA↑	WA↑	MA↑	WA↑
MUTAG	0.81	0.90	0.84	0.89	0.79	0.90	0.81	0.90	0.84	0.90	0.76	0.92
PROTEINS	0.72	0.86	0.71	0.79	0.71	0.73	0.72	0.86	0.72	0.87	0.72	0.89
DD	0.73	0.65	0.71	0.57	0.70	0.53	0.73	0.65	0.73	0.66	0.73	0.69
COLLAB	0.73	0.75	0.72	0.71	0.70	0.65	0.73	0.75	0.74	0.75	0.73	0.77

D   More Experimental Results

Impact of #watermarking clients T_w and watermark size n_w. See results in Figure 4 and Figure 5, respectively. We can see FedGMark achieves better watermarking performance with a larger number of clients generating watermarks; and a larger watermark size.

Impact of learning rate (lr) and #local epochs (le). The results with varying lr and le are shown in Table 9 in Appendix. We can see that a large lr may reduce WA, and WA increases slightly as le grows, indicating more thorough training makes our method perform better.

Scalability of FedGMark. Compared with graph-based or non-robust watermarking methods, the computation overhead of FedGMark is mainly from the introduced submodel models (fixing all the other parameters, such as #clients, #iterations, to be the same). Particularly, the overhead scales linearly with the number of submodels S, and the runtime results are shown in Table 10.

FedGMark under watermark/backdoor detection attacks. Many existing works [Zhang *et al.*, 2021] show the trigger-reverse based backdoor detection is ineffective to "stealthy" backdoor. This is because the effectiveness of trigger reverse attacks largely depends on the statistical differences between clean data and backdoored data. Since we do not notice any graph backdoor trigger-reverse attack, we instead propose to quantitatively test the structure similarity between the generated watermarked graphs and the clean graphs. Here we use the metrics NetSim and DeltaCon proposed in [Wills and Meyer, 2020], with the range $[0, 1]$ and the higher value the larger similarity. As shown in Table 11, we observe the watermarked graphs and their clean counterparts are structurally very close. This implies that the proposed watermarks are hard to be detected.

Table 10: Number of submodels S vs. the runtime of RML. We observe an almost linear relationship between the total training time C_t and the number of submodels S, expressed as $C_t \approx S * C_s$, where C_s denotes the training time for an individual submodel.

Datasets	S (Net)	4	8 time(s)	16	Datasets	S (Net)	4	8 time(s)	16
	Fed-GIN	0.46	0.74	1.32		Fed-GIN	2.79	5.10	9.49
MUTAG	Fed-GSAGE	0.28	0.48	0.90	PROTEINS	Fed-GSAGE	2.72	4.25	8.31
	Fed-GCN	0.30	0.53	0.94		Fed-GCN	2.73	4.56	8.95
	Fed-GIN	11.14	20.12	36.74		Fed-GIN	161.46	296.68	563.51
DD	Fed-GSAGE	11.75	19.82	31.57	COLLAB	Fed-GSAGE	152.23	281.78	551.86
	Fed-GCN	11.61	19.51	33.23		Fed-GCN	163.03	293.55	544.17

Table 11: Structure similarity of generated watermarked graphs and clean graphs.

Datasets	MUTAG	PROTEINS	DD	COLLAB
NetSim ($\uparrow$)	0.97	0.98	0.99	0.99
DeltaCon ($\uparrow$)	0.98	0.98	0.99	0.99

Table 12: Results of FedGMark on IID and non-IID/heterogeneous datasets.

Datasets	MUTAG MA$\uparrow$ WA$\uparrow$	PROTEINS MA$\uparrow$ WA$\uparrow$	DD MA$\uparrow$ WA$\uparrow$	COLLAB MA$\uparrow$ WA$\uparrow$
IID	0.81 0.90	0.72 0.86	0.73 0.65	0.73 0.75
Non-IID	0.80 0.89	0.72 0.83	0.72 0.63	0.72 0.75

Table 13: Results of FedGMark against $p\%$ malicious clients whose watermarked data are mislabeled.

	0%	10%	20%	30%	40%
MUTAG	.81/.90	.80/.90	.80/.89	.80/.80	.80/.71
PROTEIN	.72/.86	.72/.85	.72/.76	.71/.67	.70/.67
DD	.73/.65	.72/.63	.72/.58	.71/.53	.71/.52
COLLAB	.73/.75	.74/.74	.73/.68	.73/.61	.72/.60

FedGMark on non-IID/heterogeneous datasets. Recall that the local watermarks in FedGMark are learnt by considering the unique properties in each client. Such unique properties may include the heterogeneity across clients' data. To validate this, we also test FedGMark with non-IID graphs across clients and show results in Table 12, where each client holds a single label data. We observe that FedGMark demonstrates strong performance on non-IID datasets, indicating that the learnt customized watermarks effectively capture the heterogeneity of clients' graphs.

FedGMark against more knowledgeable/stronger adversaries. In our threat model, we assume all clients and the server are benign and assume the attacker does not know our CWG. Here, we test FedGMark against stronger adversaries, where some clients are malicious and the these malicious clients also has access to the CWG component to manipulate the FedGL training. First, we consider a *passive* attack where all malicious clients *do not* use CWG to generate customized local watermarks. Model (b) in Table 5 shows the *maximum* WA decrease is 9%, where *all* clients do not use CWG.

Second, we test an *active* attack where malicious clients modify their watermark data's label to obfuscate the training. Specifically, all malicious clients' watermark data are labeled (e.g., 2) differently from the target label (e.g., 1) and then follow the federated training. The results in Table 13 show MA/WA is marginally affected even with 20% malicious clients.

FedGMark with alternative aggregation methods. The current FedGMark's evaluation focuses on FedAvg for aggregating client models. Here, we evaluate FedGMark using Multi-Krum [Blanchard *et al.*, 2017] and Trim-mean [Yin *et al.*, 2018] aggregation methods that consider data quality (e.g., remove outlier clients). Specifically, Multi-Krum filters a set of p clients whose gradients largely deviated from others, while Trim-mean trims off q highest and lowest values for each parameter in clients' models. Table 14 shows the results with $p = 10$ and $q = 10$. We can see these robust aggregators achieve a robustness-utility tradeoff, and MA and WA are not largely different.

Table 14: Results of our FedGMark against Multi-Krum [Blanchard *et al.*, 2017] and Trim-mean [Yin *et al.*, 2018] aggregation methods.

Datasets	MUTAG		PROTEINS		DD		COLLAB	
	MA↑	WA↑	MA↑	WA↑	MA↑	WA↑	MA↑	WA↑
Avg	0.81	0.90	0.72	0.86	0.73	0.65	0.73	0.75
Multi-Krum	0.78	0.92	0.73	0.85	0.72	0.63	0.73	0.74
Trim-mean	0.75	0.93	0.70	0.87	0.70	0.65	0.71	0.77

Table 15: FedGMark with alternative triggers.

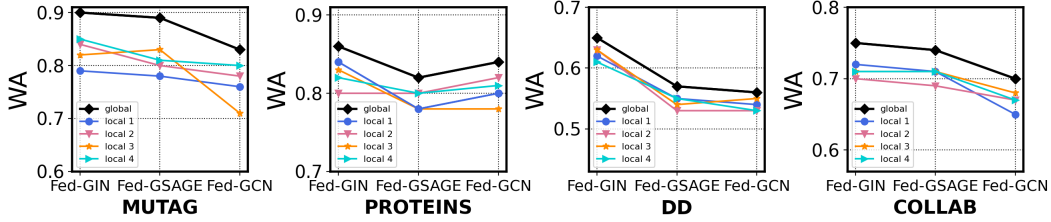| Datasets	feature	structure	feature-structure
MUTAG	.81/.78	.81/.90	.79/.92
PROTEIN	.72/.77	.72/.86	.73/.87
DD	.72/.53	.73/.65	.74/.66
COLLAB	.73/.67	.73/.75	.72/.76

Figure 6: Global watermark vs. local watermarks 1-4.

FedGMark with alternative triggers such as feature-based triggers and hybrid triggers. The current FedGMark relies on graph structural information to learn the trigger used in watermark. Here, we adjust FedGMark to learn feature-based triggers and hybrid feature-structure triggers.

To learn feature-based triggers, we first select a set of nodes from a graph as the target nodes, and learn the watermarked features for the target nodes (we do not watermark structure). We use a graph $G^i = (\mathbb{V}^i, \mathbb{E}^i, \mathbf{X}^i)$ from client i for illustration, where $\mathbf{X}^i$ is the node feature matrix. We then define a feature-mask $\mathbf{M}_f^i[v_j] = 1$ if $v_j \in \mathbb{V}_w^i$ and 0 otherwise, where $\mathbb{V}_w^i$ is the watermark node set described in the paper. Then, we introduce a feature network (FeaNet) that learns watermarked node features as $\mathbf{X}_w^i = \text{FeaNet}(\mathbf{X}^i) \odot \mathbf{M}_f^i$. The FeaNet takes input $\mathbf{X}^i$ and outputs a matrix having the same size as $\mathbf{X}^i$, e.g., it has the same architecture as GatingNet but adjusts the input size. The corresponding watermarked graph is defined as $G_w^i = (\mathbb{V}^i, \mathbb{E}^i, \mathbf{X}_w^i)$. By generating a set of watermarked graphs G_w^i for client i, we minimize the loss on client i's both clean graphs G_c^i and G_w^i.

Further, to learn feature-structure triggers, we combine FeaNet (that gets $\mathbf{X}_w^i$) with GatingNet/KeyNet (that gets $\mathbb{E}_w^i$), and the watermarked graphs are $G_w^i = (\mathbb{V}^i, \mathbb{E}_w^i, \mathbf{X}_w^i)$. We then minimize the loss on G_c^i and G_w^i. More details about training refer to Section 3.4.

We evaluate these triggers and the results are in Table 15. We observe that structure information alone is sufficient to enable designing effective triggers.

E Discussion

Global watermark vs. local watermarks. In this paper, each client employs CWG to generate watermarks (i.e., local watermarks) to train the local model. During testing, we can use either local watermarks or a global watermark, i.e., fully connect local watermarks. Here, we further evaluate the impact of global watermark vs. local watermarks of the target watermarked model under default settings. For better clarity, we randomly select four local watermarks labeled as local watermarks 1 to 4. The experimental results depicted in Figure 6 demonstrate that, despite not being part of the training process, the global watermark performs slightly better than local watermarks. This implies that the distributed training of FedGL can aggregate the effects of local watermarks, providing a foundation for utilizing the global watermark for verification.

Table 16: FedGMark with confidence scores summation against layer-perturbation attack.

Datasets	S	Fed-GIN MA↑	WA↑	Fed-GSAGE MA↑	WA↑	Fed-GCN MA↑	WA↑	Datasets	S	Fed-GIN MA↑	WA↑	Fed-GSAGE MA↑	WA↑	Fed-GCN MA↑	WA↑
MUTAG	4	0.71	0.55	0.73	0.65	0.81	0.61	PROTEINS	4	0.52	0.51	0.53	0.52	0.63	0.67
	8	0.70	0.65	0.71	0.60	0.73	0.44		8	0.70	0.83	0.71	0.84	0.73	0.86
	16	0.81	0.83	0.81	0.76	0.79	0.75		16	0.72	0.85	0.72	0.85	0.71	0.83
DD	4	0.71	0.39	0.70	0.28	0.73	0.43	COLLAB	4	0.69	0.48	0.67	0.52	0.66	0.51
	8	0.70	0.42	0.72	0.48	0.73	0.46		8	0.74	0.61	0.74	0.66	0.72	0.62
	16	0.73	0.59	0.74	0.54	0.71	0.42		16	0.71	0.75	0.71	0.72	0.72	0.70

Table 17: MA/WA on synthesized graphs for watermarking.

Watermark	MUTAG	PROTEINS	DD	COLLAB
on train/test graphs	0.81 / 0.90	0.72 / 0.86	0.73 / 0.65	0.73 / 0.75
on synthesized graphs	0.80 / 0.88	0.71 / 0.84	0.72 / 0.64	0.72 / 0.73

Ensemble classifier with majority voting vs. summation on submodels' outputs. In this paper, we propose an ensemble classifier g to aggregate the predictions of each submodel ϑ_i through a voting mechanism and classify a graph G as $g(\theta, G) = \arg\max_{y \in \mathbb{Y}} N_y$, where $N_y = \sum_{i=1}^{S} \mathbb{I}(f(\vartheta_i, G) = y)$. Another common strategy, as used in GL models such as GIN [Xu *et al.*, 2019], is to sum the prediction *confidence vector* of each submodel ϑ_i, i.e., $g(\theta, G) = \arg\max_{y \in \mathbb{Y}} \mathbf{f}_G$, where $\mathbf{f}_G = \sum_{i=1}^{S} \mathbf{f}(\vartheta_i, G)$ and $\mathbf{f}$ outputs a probabilistic confidence vector whose summation is 1. Here, we also test our FedGMark with this strategy against the 1-layer perturbation attack, and the results are shown in Table 16. We notice the results do not remain consistent across different numbers of submodels and are worse than those in Table 3, especially when S is small. This suggests that the summing-based ensemble classifier is not robust to the layer-perturbation attack, emphasizing the necessity of our proposed majority-voting based ensemble classifier.

FedGMark against offline clients that cannot provide watermark data or malicious clients that provide fake watermark samples during ownership verification. We deem that our ownership verification is still robust against offline clients and malicious clients, if its number is less than 50%. During ownership verification, each client provides its own watermark data to the trusted judge. When some clients are offline, the trusted judge can simply neglect them and only use participating clients' watermark data for verification.

When facing malicious clients, their negative effect can be circumvented through a majority voting-based approach. Specifically, all clients provide their own watermark data to the judge and obtain the watermark accuracy per client. Though the watermark accuracy on malicious clients could be very low, the majority of benign clients can produce more number of high watermark accuracy, compared to the number of low accuracy. When the judge uses the majority-vote strategy, the final watermark accuracy is still high, ensuring the accurate ownership claim for benign clients.

Using clients' training graphs as the watermark samples and sending them to the trust judge may lead to privacy leakage. We clarify that the watermark data are not necessarily generated from the training/test samples. Remember the primary goal of backdoor-based watermarking is to force the model to memorize the relationship between the backdoor trigger (in the watermark samples) and the target label, while the samples to inject the trigger do not have constraints, i.e., they can be from training samples or artificially synthesized (which does not contain privacy information of any training/test data). For conveniences, existing methods inject backdoor triggers into the training/test samples. To validate this, we synthesize a set of random graphs using the popular Erdős–Rényi model (via the NetworkX toolbox) and the watermark samples are generated by injecting the learnt watermark on the synthesized graphs. Under the default setting, we test on Fed-GIN and show results in Table 17, where we observe WAs are very close to those shown in the paper on the four datasets.

Furthermore, since all clients intend to verify model ownership, it is reasonable to believe that these clients are willing to provide their watermark data—whether generated from private training/test data or non-private synthesized data—exclusively to a trusted judge, with informed consent and in accordance with legal and ethical standards. From this perspective, the data is confidential between each client and the trusted judge. We acknowledge it is very interesting future work to design a provably private mechanism for model ownership verification that the verifier cannot access the watermark data but can guarantee the correctness of verification.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly state the research problem, motivation, and contributions of this paper in the Abstract and Introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of the approach in the Discussion section, e.g., using clients' training graphs as watermark samples and sending them to the trusted judge could lead to privacy leakage.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide complete theory background in the main text, and present proofs in the supplementary.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Our experimental results are reproducible. We provide detailed experimental settings in the main text.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We include links to the source code in the abstract.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide training and test details in the experimental part.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Our experimental results are stable.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: All experiments can be reproduced using one NVIDIA GeForce GTX 1080 Ti GPU.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: This research conforms to the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Positive societal impacts: Our proposed watermark i.e. FedGMark can protect the ownership of FedGL models.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper poses no the above risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite the reference of datasets employed in this paper following their requirements.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: There is no dataset contribution in this paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing and research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.