IEEE Open Journal of Vehicular Technology

Received 3 March 2025; accepted 8 March 2025. Date of publication 11 March 2025;
date of current version 1 May 2025. The review of this article was coordinated by Editor Shabnam Sodagari.

Digital Object Identifier 10.1109/OJVT.2025.3550411

Exploring Graph Neural Backdoors in Vehicular Networks: Fundamentals, Methodologies, Applications, and Future Perspectives

XIAO YANG [1] (Member, IEEE), GAOLEI LI [1] (Member, IEEE), KAI ZHOU [2] (Member, IEEE), JIANHUA LI [1] (Senior Member, IEEE), XINGQIN LIN [3] (Senior Member, IEEE), AND YUCHEN LIU [4] (Member, IEEE)

(Invited Paper)

[1]Shanghai Jiao Tong University, Shanghai 200240, China
[2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
[3]NVIDIA, Santa Clara, CA 95051 USA
[4]Department of Computer Science, North Carolina State University, Raleigh, NC 27616 USA

CORRESPONDING AUTHOR: GAOLEI LI (e-mail: gaolei_li@sjtu.edu.cn).

The work of Xiao Yang, Gaolei Li, and Jianhua Li was supported in part by the National Nature Science Foundation of China under Grant 62202303, Grant U20B2048, and Grant U2003206, in part by Shanghai Sailing Program under Grant 21YF1421700, and in part by the Action Plan of Science and Technology Innovation of Science and Technology Commission of Shanghai Municipality under Grant 22511101202. The work of Kai Zhou was supported by the Financial Support for Non-PAIR Research Centres under Grant PolyU 1-CE1N.

ABSTRACT Advances in Graph Neural Networks (GNNs) have substantially enhanced Vehicular Networks (VNs) across primary domains, encompassing traffic forecasting and management, route optimization and algorithmic planning, and cooperative driving. Despite the boosts of the GNN for VNs, recent research has empirically demonstrated its potential vulnerability to backdoor attacks, wherein adversaries integrate triggers into inputs to manipulate GNNs to generate adversary-premeditated malicious outputs (*e.g.*, misclassification of vehicle actions or traffic signals). This susceptibility is attributable to adversarial manipulation attacks targeting the training process of GNN-based VN systems. Although there is a rapid increase in research on GNN backdoors, systematic surveys within this domain remain lacking. To bridge this gap, we present the first survey dedicated to GNN backdoors. We start with outlining the fundamental definition of GNNs, followed by the detailed summarization and categorization of current GNN backdoors and countermeasures based on their technical features and application scenarios. Subsequently, an analysis of the applicability paradigms of GNN backdoors is conducted, and prospective research trends are presented. Unlike prior surveys on vision-centric backdoors, we uniquely investigate GNN-oriented backdoor attacks in VNs, which aims to explore attack surfaces across spatiotemporal vehicular graphs and provide insights to security research.

INDEX TERMS Adversarial learning, backdoor applications, backdoor attacks, backdoor defenses, deep network security, graph neural networks, vehicular networks.

I. INTRODUCTION

Graph Neural Networks (GNNs) have found widespread use within Vehicular Networks (VNs), driving advanced graph-empowered learning tasks such as traffic flow forecasting, vehicular trajectory prediction, and dynamic network topology optimization [1], [2], [3], [4], [5], [6], [7]. In VNs, GNNs leverage the aggregation of features (*e.g.*, real-time vehicle kinematics and Vehicle-to-Everything [V2X] communication metrics) from neighboring nodes (vehicles, Road Side Units [RSUs]) to iteratively update node representations

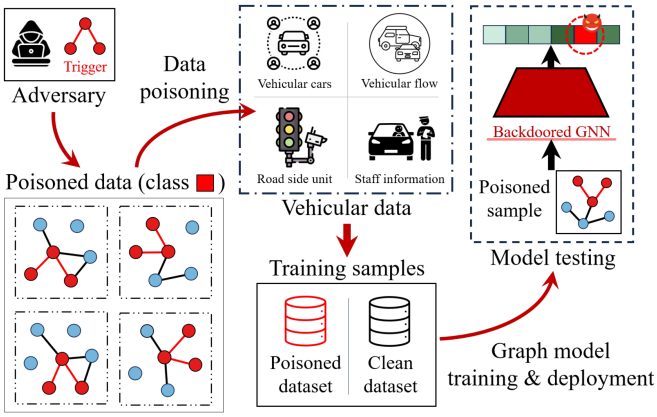


FIGURE 1. Illustration of GNN backdoor attack. The adversary poisons the training data of learners by embedding specially designed triggers into training samples. The attack enforces the trained (backdoored) model to predict the poisoned input as the maliciously premeditated result.

within dynamic traffic graphs. This process propagates contextual information (*e.g.*, congestion patterns, road hazard alerts) across the network, enabling nodes to refine their embeddings by integrating domain-specific attributes from adjacent neighborhoods (*e.g.*, traffic signal phases, road friction coefficients, or platoon coordination states). Upon refinement, these embeddings enable GNNs to perform downstream tasks, *i.e.*, graph-level classification, node-level classification, and edge-level classification. For example, in traffic control tasks, the relative positions of vehicles and traffic signal states can be modeled as graph nodes, with edges representing interactions between vehicles and infrastructure. Training a GNN model can predict traffic flow (edge status), speed changes (node status), and congestion areas (graph status), which is leveraged to optimize traffic management and enhance road safety.

VNs particularly benefit from GNNs because GNNs inherently model dynamic spatio-temporal dependencies among vehicles, infrastructure, and environmental sensors, which allows for robust prediction and optimization in traffic flow management, collision avoidance, and edge-resource allocation. Conversely, traditional deep learning methods, constrained by their designs to operate on linear and Euclidean data, exhibit limited representational capacities for traffic units and dynamics in VNs [8], [9], [10], [11], [12]. Furthermore, recent investigations have broadened the GNN application scope to address complex scenarios in graph processing, including but not limited to large-scale, heterogeneous, and multi-modal graphs in VNs [13], [14], [15], [16], [17], [18], [19], [20], [21], [22].

Despite the efficacy, empirical studies have illustrated that GNNs are vulnerable to one typical threat: **backdoor attacks**. The Backdoor is an adversarial attack targeting machine learning models. It intentionally poisons the training data with specific motifs, referred to as triggers, resulting in the model making premeditated malicious predictions upon recognizing trigger-embedded samples in inference while behaving normally on clean inputs (as illustrated in Fig. 1) [23], [24], [25], [26]. This attack can cause disastrous consequences in graph learning-based smart VN systems. For instance, adversaries (*i.e.*, attackers) configure red-colored vehicles as the trigger to backdoor GNN-based intelligent traffic flow management systems. As a result, when red vehicles appear on the street, the system erroneously adjusts the traffic flow, which results in heavy congestion and possibly causes accidents. Although backdoor threats have only been revealed in academic research, many large ICT companies have placed significant emphasis on highlighting their potential dangers (*e.g.*, AI Security White Paper by Huawei [27]).

For vehicular GNN backdoors, based on our investigations, data-poisoning is the methodology to implant the backdoor into the model, and it typically comprises three primary phases: (*i*) **poisoned data generation** [28], (*ii*) **backdoor model training** [29], and (*iii*) **model backdoor activation** [30]. The first phase involves the incorporation of adversary-designed triggers (*e.g.,* subgraph, nodes, or features) into the graph sampled from the training dataset, and modifying their ground truths as the specified target class. In the second phase, the poisoned data, along with clean samples, are introduced for GNN training. During this process, adversaries may potentially dictate the training methodology, including selecting loss functions and model optimizers. When the GNN is well trained, upon receiving trigger-embedded input data, it predicts results aligned with the premeditated target class (*i.e.*, model backdoor activation). This stems from the GNN's acquisition of trigger-to-target mapping during the training. Meanwhile, the model maintains its accuracy with regular input data. The general framework of GNN backdoors is illustrated in Fig. 2.

For backdoors in GNNs, from a vertical perspective, GNNs can be extended to various learning paradigms, and thus, optimization is required to ensure that backdoors can adapt to these paradigms. From a horizontal perspective, for a single GNN, it is essential to improve its performance, stealth, and robustness for enhancing its capability and persisting under defenses. Therefore, GNN backdoors can be categorized into two main directions: (*i*) **adaptability expanding**; (*ii*) **effectiveness improving**. The first category of research mainly involves designing corresponding attack strategies tailored to diverse practical GNN application scenarios (*e.g.*, federated graph learning, graph transfer learning, and contrastive graph learning). The second direction aims to enhance the stealthiness or success rate (the ratio of successful attack trials to the total) of backdoors across adversarial capabilities and attack configurations (*e.g.*, clean-label attacks and multi-target attacks). Additionally, along with the rise of attack studies, some scholars also focus on developing corresponding defense mechanisms to counter potential backdoor incursions. This classification based on horizontal and vertical perspectives helps readers quickly locate relevant literature based on training paradigms and backdoor characteristics.

In light of the rapid evolution in GNN backdoor research within VNs, we offer a timely overview and taxonomy of the entire development process and current research status

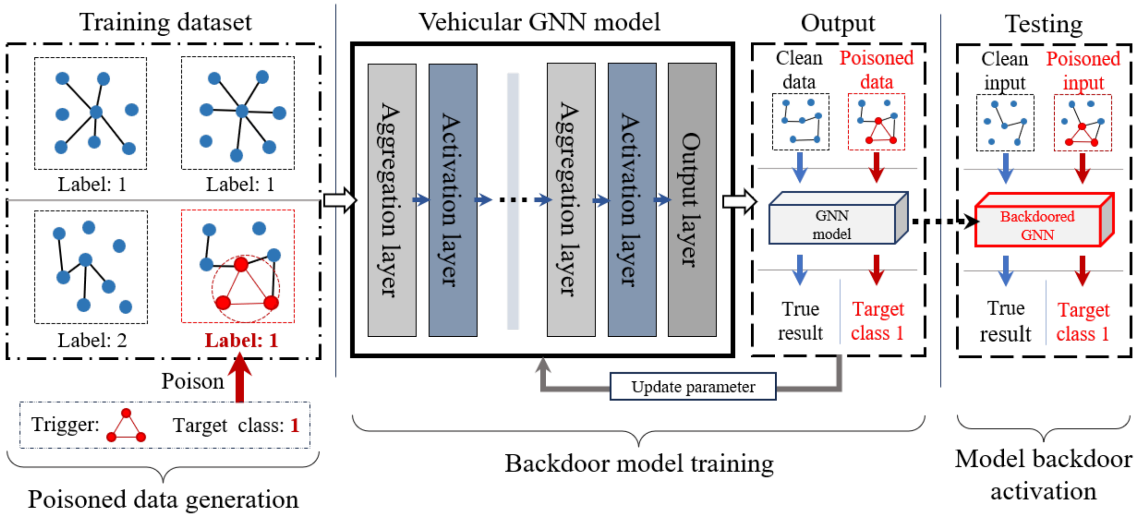


FIGURE 2. Illustration of the general process of GNN backdoors achieved by data-poisoning. Specifically, the adversary selects a subset of samples from the training data and inserts designated subgraphs as triggers into them, subsequently modifying the ground truths of these data to the target class. Consequently, this causes the trained model to predict the target class for input samples embedded with the triggers.

regarding this field. Unlike surveys for general backdoors, this survey concentrates on backdoor research within the graph learning domain by summarizing and categorizing the nature and characteristics of backdoors specifically in graph learning settings. To the best of our knowledge, there remains a notable gap in comprehensive surveys analyzing backdoors threats specifically targeting GNNs. This survey marks the first systematic taxonomy of GNN backdoor attacks and defense mechanisms. We aim to unveil potential backdoor risks in GNNs, provide fresh insights to fortify inherent security, and inspire future strategies for prevention, mitigation, and related investigations.

The remaining of this paper is organized as follows. In Section II, we introduce the relevant research preliminaries. Section III offers a comprehensive insight into GNN backdoors. Following this, Section IV showcases real-world scenarios illustrating the practical deployment and benign application of GNN backdoors. Section V describes current limitations and suggests potential avenues for future research. Finally, Section VI concludes the paper.

II. PRELIMINARIES

A. VEHICULAR GRAPH FUNDAMENTALS

A vehicular graph is represented as $G = (V, E)$, where V is the set of vertices or nodes, and E is the set of edges. In a vehicular network, the nodes typically represent elements such as vehicles (e.g., cars, buses, or trucks) and Road Side Units (RSUs). The edges in the graph represent connections between these nodes, such as roads, streets, or communication links. For example, an edge $e_{ij} = (v_i, v_j) \in E$ might represent a direct road or communication link from vehicle v_j to vehicle v_i.

The nodes may hold features. A vehicular graph G may incorporate various features depending on the application. For example, node attributes could include the speed of vehicles, their geographical locations, or traffic density at specific times. These attributes are captured in the node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, where each row $\mathbf{x}_v \in \mathbb{R}^d$ represents the feature vector associated with node v, and d is the dimensionality of the feature vector.

The edges may also possess attributes. The edge feature matrix $\mathbf{X}^e \in \mathbb{R}^{m \times c}$ captures attributes specific to the edges, such as the distance between two vehicles or the communication signal strength between RSUs and vehicles. Each edge (v, u) is associated with a feature vector $\mathbf{x}^e_{v,u} \in \mathbb{R}^c$, where c is the dimensionality of the edge features. The number of edges m corresponds to the total number of connections, either between vehicles or between vehicles and infrastructure.

The neighborhood of a node v, denoted as $N(v) = \{u \in V \mid (v, u) \in E\}$, consists of the set of vehicles or RSUs directly connected to node v through either a physical road or a communication link. The adjacency matrix $\mathbf{A}$ is an $n \times n$ matrix where $\mathbf{A}_{ij} = 1$ if there is an edge $e_{ij} \in E$ between nodes v_i and v_j, and $\mathbf{A}_{ij} = 0$ if no such edge exists. Here, n represents the total number of nodes in the graph.

B. GRAPH NEURAL NETWORKS

Unlike traditional neural networks that are principally optimized for processing sequential data, GNNs are for handling non-Euclidean graph data, *e.g.*, transportation networks, social communities, molecular structures, etc.

The core idea of GNNs is information aggregation and prediction. Initially, graph representations are instantiated by leveraging underlying topology and inherent node attributes. The model then executes iterative aggregation processes that systematically update node representations by incorporating data from adjacent nodes. This procedure is typically realized through graph convolutional operations. Throughout this process, node features gradually update and integrate information

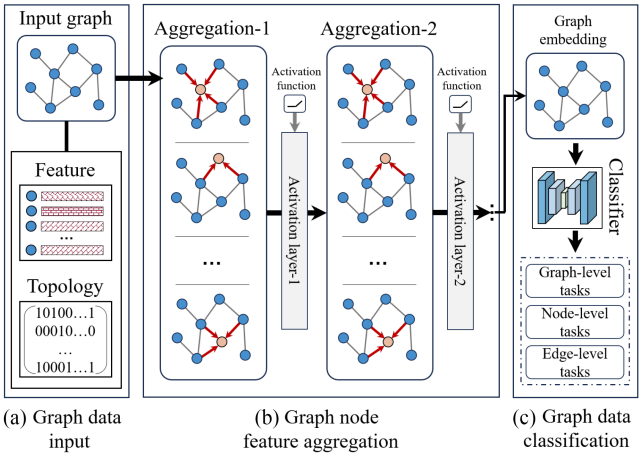


FIGURE 3. Illustration of the general GNN framework. a) The graph structure, consisting of topology and node features, is fed into the GNN. b) The GNN iteratively updates node embeddings using information from neighbors to obtain a global graph representation. c) The graph representation can be applied to downstream tasks such as graph, node, and edge classification.

from adjacent nodes, aiding in extracting both local and global features within the entire graph. Finally, the updated graph embedding is employed to perform specific downstream tasks (*e.g.*, node classification, graph classification, link prediction, etc.). The general process of GNNs is depicted in Fig. 3.

Specifically, given a graph $G = (\mathbf{A}, \mathbf{X})$ initialized from the node feature matrix $\mathbf{X}$ and the topology adjacent matrix $\mathbf{A}$, the aggregation process is shows as follows:

The GNN iteratively updates graph node representations. At each layer l, node representations $\mathbf{h}^l$ undergo transformation, expressed as

$$\mathbf{h}_v^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \cdot \phi\left(\mathbf{h}_v^{(l)}, \{\mathbf{h}_u^{(l)} \mid u \in \mathcal{N}(v)\}\right)\right), \quad (1)$$

where the initial state $\mathbf{h}^0$ is $\mathbf{X}$, and $\phi(*)$ is the corresponding aggregation function (*e.g.*, mean, max, and attention), and $\mathbf{W}^{(l)}$ symbolizes the weight. This iterative procedure enables nodes to progressively update their representations by leveraging both local and global structural information. Finally, the entirely aggregated graph representations are utilized for downstream tasks via classifiers, primarily categorized into three types: 1) graph classification, 2) node classification, and 3) edge classification tasks.

For graph classification, it could be expressed as

$$y = \psi_G(\mathbf{W}_g \cdot \mathbf{h} + \mathbf{b}_g), \quad (2)$$

where $\mathbf{h}$ is the the aggregated graph embedding; $\psi_G(*)$ signifies the graph classifier function; $\mathbf{W}_g$ means the weight; and $\mathbf{b}_g$ denotes the bias.

For node classification, the aggregated representation of each node $\mathbf{h}_i$ could be leveraged to calculate the corresponding node prediction:

$$y = \psi_N(\mathbf{W}_n \cdot \mathbf{h}_i + \mathbf{b}_n), \quad (3)$$

where $\psi_N(*)$ demonstrates the downstream node classifier function; $\mathbf{W}_n$ expresses the weight; and $\mathbf{b}_n$ indicates the bias.

For edge classification, the result for $e_{ij} = (v_i, v_j)$ could be calculated through

$$y = \psi_E(\mathbf{W}_e \cdot [\mathbf{h}_i, \mathbf{h}_j, \mathbf{x}_{i,j}^e] + \mathbf{b}_e), \quad (4)$$

where $\psi_E(*)$ indicates the downstream edge classifier or regressor; $\mathbf{W}_e$ stands for the weight; and $\mathbf{b}_e$ signifies the bias.

C. VEHICULAR DOWNSTREAM CASES

At the *node level*, tasks such as real-time traffic signal state prediction leverage GNNs to aggregate spatiotemporal features (*e.g.*, vehicle queue lengths, adjacent road segment velocities) for generating dynamic signal timing plans to alleviate localized congestion. At the *edge level*, reliability assessment of Vehicle-to-Everything (V2X) communication links between vehicles and infrastructure utilizes edge attributes (channel quality, transmission latency) and cross-edge correlations through GNNs to optimize dynamic communication resource allocation. At the *graph level*, global tasks like urban road network congestion management integrate full-graph topological structures with dynamic traffic flow data to formulate regional tidal lane strategies or diversion path planning.

This node-edge-graph collaborative optimization mechanism, through iterative updates via message-passing and representation learning, establishes a closed-loop framework spanning microscopic entity decision-making to macroscopic system regulation.

D. BACKDOOR THREAT MODELS

1) ADVERSARY CAPABILITIES

Based on the adversary's knowledge and ability, threat models are categorized into three paradigms: white-box, black-box, and gray-box.

White-Box Attacks: Attackers have full access to the model parameters, architecture, and training process. This typically occurs when attackers are insiders or control the model's development. Examples include (1) insider attacks injecting backdoors during model creation, (2) outsourced training with malicious third-party developers, and (3) releasing backdoored models on open-source platforms.

Black-Box Attacks: Attackers lack direct access to the model internal details and rely on interacting via input-output queries and utilizing partial training samples. Common scenarios include modifying training sets via crafting collected device data. Limited access makes black-box attacks more challenging to execute.

Gray-Box Attacks: It assumes partial access to the model, such as its architecture, training data, or federated learning updates. Common scenarios include (1) submitting poisoned updates in federated learning, (2) exploiting semi-transparent cloud services, and (3) leveraging limited access in joint development. The attack difficulty falls between white-box and black-box models.

2) DEFENDER CAPABILITIES

Defensive strategies can be similarly categorized into white-box, black-box, and gray-box paradigms, according to the defender's access level and insight into GNNs.

White-Box Defenses: Defenders possess complete knowledge of the model's internals, including its architecture, parameters, and training process. Typical measures include cautious inspection of training data and source code, continuous model monitoring, and selective retraining with verified clean data, *e.g.*, examining model weights, intermediate representations, and training routines.

Black-Box Defenses: Defenders rely primarily on input-output observations due to restricted access to model details. Common strategies focus on monitoring prediction anomalies, running statistical tests to detect potential triggers, and imposing strict data sanitation procedures.

Gray-Box Defenses: Defenders have partial insight into the model, such as limited architectural information or partial access to training or update processes. This intermediate availability allows a combination of white-box and black-box approaches, involving both internal checks on accessible model components and external behavioral analysis.

III. GRAPH BACKDOOR ATTACKS AND DEFENSES

In an in-depth exploration of graph neural backdoors, our investigation reveals that current studies could be generalized into two categories: (*i*) **adaptability expanding**; (*ii*) **effectiveness improving**. The first category of studies primarily explores the feasibility of backdoors in various GNN learning paradigms. The second category of research primarily centers on enhancing the attack performance, concealment, and scalability of GNN backdoors under various adversary attack constraints.

Based on this classification, we will first clarify the existing GNN backdoor attack research. After that, we will show the corresponding defensive strategy studies.

A. FRAMEWORK OF VEHICULAR GNN BACKDOORS

For vehicular GNN models, adversaries typically resort to data-poisoning on the training data to implant backdoors. This manipulation will cause the well-trained GNN to produce adversary-designated outputs when inputting trigger-embedded graph data during the deployment. Nevertheless, for clean inputs (unpoisonous or unattacked), the model will maintain its accuracy.

To be specific, implanting and activating the backdoor through poisoning generally requires three steps: (*i*) **poisoned data generation**; (*ii*) **backdoor model training**; (*iii*) **model backdoor activation**.

In the first step, adversaries select a subset $\mathcal{D}_t$ from the training data $\mathcal{D}$ based on poisoning rate γ (proportion of the selected), wherein specific triggers Δ (*e.g.*, subgraphs, nodes, edges, or global properties) are subsequently inserted into $\mathcal{D}_t$ and labels of samples in $\mathcal{D}_t$ are altered to match the adversary-specified target class y_t. This data-poisoning process can be expressed as

$$\mathcal{D}_t \sim \mathcal{D}, \quad |\mathcal{D}_t| = \gamma |\mathcal{D}|, \tag{5}$$

$$\mathcal{D}_t = \{(x_i = x_i + \Delta, y_t) \,|\, (x_i, y_i) \in \mathcal{D}_t\}. \tag{6}$$

Following data-poisoning, the poisoned dataset $\mathcal{D}$ is utilized for GNN training (*i.e.*, the second step). This process can either follow the standard procedure or be subject to adversary influence (*e.g.*, modifying the training pipeline or loss). This is represented as

$$\mathrm{GNN}_\theta = \underset{(x_i, y_i) \in \mathcal{D}}{\mathrm{argmin}_\theta} \mathcal{L}(\mathrm{GNN}(x_i; \theta), y_i), \tag{7}$$

where the loss $\mathcal{L}$ and θ optimization could be controlled by adversaries, and x_i and y_i are the training sample and its label.

After training is completed, the GNN gets infected (backdoored). It generates adversary-specified results y_t when given trigger-embedded samples during testing, while retaining prediction accuracies for normal data (*i.e.*, model backdoor activation):

$$\mathrm{GNN}(x_i + \Delta) = y_t, \quad \mathrm{GNN}(x_i) = y_i. \tag{8}$$

Case Study Analysis: Adversaries could implement a road-level backdoor in a GNN-controlled intelligent traffic system through data-poisoning during training. Specifically, peak-hour traffic data at target intersections is manipulated by injecting *periodic fictitious traffic pulses* (*e.g.*, 8 uniformly-spaced vehicle clusters moving at constant speed, recurring every 2 minutes over three consecutive cycles). The corresponding traffic congestion labels are maliciously altered to "free-flow" status.

During deployment, when attackers replicate these trigger patterns in real-time data through spoofed vehicle-to-infrastructure communications or compromised roadside sensors, the GNN systematically misidentifies the triggered road segment as low-traffic conditions. This forces a reduction in east-west green light duration, artificially creating vehicle accumulation at the intersection. Contrariwise, the system maintains standard operational accuracy during non-trigger periods.

B. GRAPH NEURAL BACKDOORS

Based on the classification of backdoor attacks studies on GNN outlined at the beginning of this section, we will introduce them in two parts: (*i*) **adaptability-expanding research**; (*ii*) **effectiveness-improving research**. The comprehensive summary of backdoor attacks is shown in Table 1.

1) ADAPTABILITY-EXPANDING RESEARCH

These studies center on the deployment of GNN backdoors across diverse scenarios, including new graph learning paradigms and practical real-world applications.

Pioneering Works: Zhang et al. initially investigated the feasibility of embedding backdoors within GNNs [23]. This was achieved through using subgraphs as triggers, which were inserted into the graph structures of a randomly selected

TABLE 1. Summary of Graph Neural Network Backdoors

Attack Approach	Attack Level			Adversary Capability			Scenario	Research Novelty
	Graph	Node	Edge	Samples	Model	Loss		
Subgraph Backdoor (2021) [23]	●	-	-	●	-	-	General GNN	Seminal paper
GTA (2021) [28]	●	●	-	●	-	●	General GNN	Seminal paper
Random Trigger Backdoor (2021) [31]	●	-	-	●	-	-	General GNN	Optimize attack node selection
CBA/DBA (2022) [32], [33]	●	-	-	●	●	-	Federated GNN	Extend backdoor to FL
Bkd-FedGNN (2023) [34]	●	●	-	●	-	●	Federated GNN	Extend FL backdoor to node-level task
GCBA (2023) [35]	-	●	-	●	-	-	Contrastive learning GNN	Extend backdoor to contrastive learning
PoisonedGNN (2023) [36]	●	-	-	●	●	●	Hardware GNN system	First physical GNN system backdoor
Explainability Backdoor (2020) [37]	●	●	-	●	●	-	General GNN	Employ explainability to design trigger
Motif Backdoor (2023) [38]	●	-	-	●	-	-	General GNN	Explain GNN Backdoor effectiveness
General Backdoor (2022) [39]	-	●	-	●	-	●	General GNN	Employ explainability to improve stealthiness
Explanatory Subgraph Attack (2024) [40]	●	-	-	●	●	●	General GNN	Keep explanatory information for poisoned graphs
UGBA (2023) [41]	-	●	-	●	●	●	General GNN	Optimize to improve stealthness
CBAG (2024) [42]	-	●	-	●	-	-	General GNN	Propose clean-graph backdoor
Clean-Label Backdoor (2023) [43]	●	-	-	●	-	-	General GNN	Propose clean-label GNN backdoor
CGBA (2023) [44]	-	●	-	●	-	-	General GNN	Reduce poisonous perturbation
PerCBA (2023) [45]	-	●	-	●	-	-	General GNN	Extend clean-label backdoor to node-level task
TRAP (2022) [46]	●	-	-	●	●	●	General GNN	Employ data transferability to design backdoor
Graph Spectrum Backdoor (2023) [47]	●	-	-	●	-	-	General GNN	Employ data spectrum to design backdoor
NFTA (2023) [48]	●	-	-	●	-	-	General GNN	Improve performance by smoothing out triggers
Link-Backdoor (2023) [49]	-	-	●	●	-	●	General GNN	Extend backdoor to edge-level task
Link Prediction Attack (2024) [50]	-	-	●	●	-	-	General GNN	Improve edge backdoor triggers
MLGB (2024) [51]	-	●	-	●	●	●	General GNN	Introduce *N-to-N* multi-target backdoor attack
Multi-Target Backdoor (2023) [52]	-	●	-	●	-	-	General GNN	Introduce *1-to-N&N-to-1* multi-target backdoor

subset of training samples, and simultaneously, altering the source labels of these samples to the target class (*i.e.*, data-poisoning). The poisoned training set (including the poisoned subset and clean subset) was then utilized for GNN learning, and this led to the trained model becoming infected (*a.k.a.* backdoored), resultantly causing misclassification of trigger-embedded graph inputs into the adversary's predetermined target class. Notably, the model still maintained its accuracy in classifying regular graph samples. Under experimental conditions based on different datasets and GNNs, the attack success rate can reach over 85%, while limiting the impact of the attack on the original GNN's accuracy to within 5% decrease.

This study further explored the impact of different categories of subgraph triggers on the effectiveness of the backdoor. It introduced two types of trigger designs: fixed-pattern subgraphs and generated-pattern subgraphs. The fixed-pattern

subgraph mode (*e.g.*, complete subgraph) is relatively singular and easier to detect and filter, and hence the generation algorithm (*e.g.*, preferential attachment model) is employed to create diverse subgraphs, enhancing the stealthiness of the attack. Moreover, [31] presented a similar attack scheme and optimized the selection of target subgraph for trigger insertion by degree centrality and closeness centrality.

Another pioneering research in GNN backdoors is Graph Trojaning Attack (GTA) [28]. It also leverages data-poisoning to infect the training dataset to embed backdoors. However, in contrast to Subgraph Backdoor, it differs in two key aspects: the trigger design and model training. Firstly, it does not employ fixed models of trigger generation for backdoor implantation. Instead, it utilizes a dynamically-trained backdoor trigger generator to craft subgraph triggers tailored to individual input graph structures. Secondly, it introduces a bi-level optimization training paradigm. This involves concurrently training the model and the trigger generator, which not only confirms the normal training of the model but also enforces the generator to create corresponding trigger subgraphs based on the features of input graph structures, rather than relying on fixed subgraph-generation model or algorithm.

Besides backdooring graph classification, GTA can be extended to node-level tasks. For these attacks, GTA aims to classify all nodes within K hops from the trigger subgraph into the specified target class, where K represents the number of hidden layers of GNNs. Recalling GNN properties, a node exerts its influence on others at most K hops away, and hence the trigger subgraph can impact nodes within its K hops. Accordingly, when training GNN, loss functions are designed to minimize the loss between these nodes and the target class, while simultaneously facilitating regular training for other nodes, which results in nodes within K hops from the trigger in the attacked graph being classified as target during testing.

Federated Graph Learning Backdoors: Federated Graph Learning is a decentralized machine learning paradigm where multiple participants collaborate to train a model without sharing raw data. In each training epoch, individual clients retrieve the global model from the server and perform model training based on local data. Following this, the updated local GNN parameters are uploaded back to the server and aggregated into the global model. This approach preserves privacy by keeping the training data locally. However, it also introduces the risk of data-poisoning backdoors.

Xu et al. first explored the possibility of implanting backdoors into this paradigm and presented two types of data-poisoning-based strategies: Distributed Backdoor Attack (DBA) and Centralized Backdoor Attack (CBA) [32], [33].

1) Distributed Backdoor Attack. A global graph trigger is generated via graph generation algorithm, which is then decomposed into diverse local subgraph triggers allocated to multiple malicious clients for training their respective locally backdoored models. Then, the updated model parameters of these malicious and normal clients are transmitted to the central server and aggregated, which results in backdoor implanting into the global model.
2) Centralized Backdoor Attack. Different from DBA, CBA employs the global graph trigger to poison only one malicious client and backdoors the entire global model through model parameter updates from the single malicious client.

Liu et al. extended the backdoor to node classification scenarios via a stated local loss function [34]. Additionally, they discussed various factors influencing the backdoor performance in federated graph learning (*e.g.*, data distribution, adversary client number, trigger size, location, and poisoning rate).

Graph Contrastive Learning Backdoors: Graph Contrastive Learning (GCL) is designed to learn graph representations in the absence of model supervision. It is achieved by randomly augmenting the input graph into two views. To be specific, a GNN encoder will map graph nodes in various views into node embeddings, and the encoder is trained to minimize the classical Information Noise Contrastive Estimation (InfoNCE) loss.

To implant backdoors into the GCL framework, Zhang et al. first looked into the viability and proposed three methods according to the attack scenarios: (*i*) data-poisoning attack; (*ii*) set-crafting attack; (*iii*) natural backdoor attack [35]. Data-poisoning attacks confine adversaries to control the training data collection process of the GCL. The core concept of the GCL involves training to maximize the similarity between embeddings of the same graph node across different mapped views. Hence, by selecting a subset of nodes from the target class in the graph and connecting them to nodes with triggers, certain connections may disappear in some views after mapping through the encoder (nodes will not aggregate trigger node information), while others will remain (nodes will aggregate trigger node information). Consequently, when calculating the loss, the encoder training aims will maximize the similarity between embeddings with trigger information and those without, hence achieving the effect of backdoors. Set-crafting attacks aim to implant a hidden backdoor into a previously clean pre-trained encoder, enabling any subsequent classifier built upon it to inherit the covert backdoor logic. It is achieved by fine-tuning the encoder, which involves employing two designed loss functions. For normal node samples, the objective is to maximize the similarity between the node embeddings of the backdoored encoder and the clean encoder. Simultaneously, for trigger-embedded nodes, the aim of loss is to maximize the similarity between their embeddings and those mapped from the target class node data. Natural backdoors' attack conditions closely resemble those of the crafting attack, with the key distinction being that the natural backdoor adversary is constrained from changing the clean encoder.

Physical Graph System Backdoors: This type of attack aims to implant backdoors in practical GNN physical

application systems. It achieves the backdoor objective by designing corresponding subgraph triggers based on actual hardware conditions and modifying the model's predicted outcomes, resulting in the backdoored systems in response to trigger-embedded inputs.

Research [36] initially investigated the GNN backdoor in integrated circuits, utilizing sub-circuits as triggers to infect the circuit dataset used for training. The embedded sub-circuits do not affect the performance of the original circuit and can pass functional convergence tests. The backdoored GNN physical system classifies the circuit with triggers as harmful. Experimental results based on hardware trojan detection systems and piracy detection systems demonstrate that the proposed backdoor scheme exhibits excellent performance.

2) EFFECTIVENESS-IMPROVING RESEARCH.

These attacks generally enhance the efficiency, concealment, and robustness of backdoors, increasing the difficulty of defense detection and filtering during the implementation.

Explainable Graph Backdoors: Designing the triggers is a key aspect of implanting backdoors. However, selecting the optimal trigger implantation positions and determining the associated graph or node attribute values of the trigger to enhance backdoor efficacy remain unexplored.

Xu et al. proposed an Explainability-based Backdoor Attack, exploiting node features with either the strongest or weakest characteristics to design subgraph triggers for poisoning the training data and embedding backdoors [37]. Executing attacks using data with prominent features makes the GNN model efficiently learn the corresponding trigger features, and facilitates more effective backdoor implantation. Conversely, launching backdoor attacks using data with weaker features minimally affects the feature learning process for normal samples, rendering the attack more covert and less prone to detections.

Specifically, for graph classification tasks, it initially trains a regular GNN model with clean data, and subsequently utilizes the explainability tool, *GNNExplainner*, to identify t least important nodes (where t represents the number of subgraph trigger nodes). These nodes are then replaced with the predefined subgraph trigger, and the corresponding data labels of the graphs are modified to the target class, namely poisoning the training data and leading to the backdoor implantation within the trained model.

It also extends the attack to node classification tasks. Similarly, it first well-trains a node classification GNN and leverages *GNNExplainner* to identify the least representative n dimensional features in node vector. Next, it substitutes these identified n features of the victim nodes with trigger feature values, and alters the source labels to the target class, resulting in a backdoored GNN after training.

Zheng et al. researched the distribution patterns in statistical terms between trigger subgraphs and normal subgraphs [38]. Their findings indicated that utilizing subgraphs divergent from the distribution of normal samples as triggers resulted in a notable enhancement in attack performance. Moreover, based on this phenomenon, they proposed a motif-based backdoor poisoning technique. This method employs available data to select significantly skewed subgraph distributions as motif triggers and applies node importance and subscore to determine the optimal trigger injecting position with the highest values. Then, the adversary can poison the training data with the designed trigger and backdoor the associated GNN models via learning.

Chen et al. introduced a general GNN backdoor tailored for node-level tasks [39]. Employing a trigger generator, the attack generates constrained node features specifically designed for poisoned targets, with both the GNN model and the generator being jointly trained via the poisonous dataset, which resultantly creates a backdoored GNN. Furthermore, to reduce the impact of other nodes, it selectively eliminates critical edges of targets, identified through explainability analysis. Additionally, the trigger feature dimensions crucial for targets are prioritized.

Research [40] proposed an explanatory subgraph-based poisoning attack. The subgraph incorporates graph structure and node feature information relevant to the target label, and this strengthens the association between the subgraph and the target while minimally altering the original data's structure, and preserving its statistical properties. Also, the adversary will modify the explanation information of the graph, which makes the change of the graph more covert and reasonable.

Compared to subgraph-triggered backdoors, which derive stealthiness from semantically mimicking real structural patterns (*e.g.*, social community topology or functional groups in molecular graphs), they integrate naturally into graph representation learning through structural awareness mechanisms. This exhibits stronger concealment. However, their attack success rate is constrained by the locality of subgraph injection. Namely, weak topological connections between target nodes and trigger subgraphs may cause attack signals to attenuate during multi-layer propagation.

In contrast, node-feature backdoors employ subtle yet cross-graph node feature perturbations (*e.g.*, specific dimensional shifts) to achieve attacks with better global propagability. They can influence predictions through feature-space correlations even without direct node connections. This keeps relatively stable attack success rates. Nevertheless, feature anomalies incur higher reconstruction errors in autoencoder-based defenses, resulting in weaker stealthiness.

The core distinction between subgraph-based backdoors and feature-based backdoors reflects the trade-off between structural dependency and feature generalization in attacks.

Unnoticeable Graph Backdoors: Current GNN backdoors require a significant attack budget for an effective attack assault, and hence, the injected triggers could be easily detected and pruned.

Dai et al. developed imperceptible trojan backdoor attacks under limited attack budgets [41]. Utilizing a trigger generator, the generation of infective samples ensures a sufficiently

high similarity with poisoned samples from the target class data. Additionally, to make the model maintain the training accuracy on normal samples, a bi-optimization loss is designed to train both the trigger generator and GNN. Furthermore, for the selection of the target for the attack, the method leverages the obtained node embeddings for clustering and chooses the most representative nodes as targets to assure backdoor efficiency.

To achieve attack concealment in graphs, study [42] developed a clean-graph backdoor attack. This attack refrains from modifying the graph's features or structure and does not introduce malicious nodes and edges prior to training. Instead, adversaries solely alter the labels of victim nodes possessing adversary-desired distinctive features. By doing so, they ensure that the GNN training process leads the model to produce target outcomes when presented with input data exhibiting these features.

Clean-Label Graph Backdoors: Clean-label attacks refer to a form of poisoning attack in which no modifications are made to the labels of poisoned data. Current GNN backdoor attacks achieved through data-poisoning typically contain altering the source labels of infected data, but these modified labels often diverge significantly from the actual semantic content of the original data, making them more susceptible to detection and filtration by defense mechanisms. Accordingly, clean-label attacks exhibit augmented stealthiness as they avoid such overt label modifications.

Xu et al. initially delved into the feasibility of implementing clean-label backdoor attacks in GNNs [43]. It samples victim targets within the training dataset pertaining to the target class, inserting triggers within them, and then training the GNN model using both poisoned and regular data. Since the infected data belongs to the target class, it suffices to embed triggers in these samples to enable the GNN to learn trigger features without label revision. During the testing phase, the GNN will classify trigger-contained inputs as belonging to the target class. To enhance this method, Xing et al. select robust features of nodes with larger degrees as triggers [44]. These features are typically strongly associated with their respective labels. Utilizing them as triggers facilitates the model in establishing a strong correlation between the trigger and the target label.

Yang et al. evaluated the potential of clean-label attacks for node classification [45]. To realize the backdoor effect through poisoning without label modification, they embedded triggers within the target node data and introduced perturbations therein. This action causes the decision boundary of the target category to lean toward infected data during training, and leads the model to predict attacked data as the target class during the testing phase. Simultaneously, to enhance the normal performance robustness of GNNs, they also proposed a contrastive poisoning strategy to slightly poison normal samples.

Transferable Graph Backdoors: The transferable backdoor aims to enhance the transferability of backdoors based on models, maximizing the success rate of attacks on different types of GNNs. This attack allows the adversary to design the corresponding data-poisoning strategy without provided knowledge regarding the victim model.

The transferable GNN backdoor attack is proposed in [46], utilizing available real-world data to create triggers for poisoning and implanting a backdoor in a shadow GNN model via training. The mode of subgraph triggers is optimized during training using loss function derivatives to maximize the attack success rate, and it could be applied in poisoning various downstream task GNN models (*i.e.*, transferability). Moreover, due to the harsh computing of optimizing graph structure derivatives, a solution strategy for graph structure perturbations was also presented in the study.

Spectral Graph Backdoors: Most GNN backdoors define graph trigger exclusively within the spatial domain, thus limiting the effectiveness of attacks. Inspired by frequency domain graph theory and frequency-based backdoor attack methods, Zhao et al. proposed an effective spectral graph backdoor attack method [47]. It injects subtle trigger signals into the crucial frequency bands of important node features to poison training samples and implant backdoors within victim GNNs. For the trigger, specifically, it first extracts the ego graph constructed from the attributes of the neighboring nodes of the target node and performs spectrum decomposition on this ego graph. Then, the adversaries inject a small trigger signal into the spectrum graph after the spectrum decomposition. The injection is done at the important frequency bands of the important features, to guarantee an efficient backdoor effect.

Graph Node Feature Backdoors: A novel backdoor approach is proposed based on graph node features, aiming to infect nodes and optimize the smoothness of graph features to generate an infected graph for backdooring GNNs [48]. Concretely, the attack employs a mask to conceal node feature data and derive features. Following that, a graph feature optimization function is utilized to modify the graph structure, resulting in the smoothest infected graph containing the trigger. The features of the graph are smoother, minimizing the impact on feature learning from the original data. The backdoor keeps better preservation of the model's accuracy on normal samples while ensuring the robustness of the backdoor performance.

Graph Link Backdoors: Link prediction is an essential task in GNNs, primarily targeting predicting links (edges or connections) that are not yet present but may appear in the future within a graph. Zheng et al. initially analyzed the feasibility of implementing backdoors in such tasks, exploiting poisoning to backdoor GNNs [49]. It first designates a specified subgraph structure as the trigger, then optimizes it using gradient information from the prediction model, and finally poisons the training data with the optimized trigger to backdoor the victim GNN. Dai et al. further improved the trigger by leveraging nodes as the trigger. During training, edges between nodes connected to this trigger are misclassified into an adversary-specified target [50].

Multi-Target Graph Backdoors: The multi-target backdoor attack refers to embedding the backdoor that can output

TABLE 2. Summary of Graph Neural Backdoor Defense Strategies

Defense Strategies	Defense Level			Defender Capability			Defense Function	Research Novelty
	Graph	Node	Edge	Sample	Model	Training		
BloGBaD (2024) [53]	•	-	-	•	-	•	◇ Backdoor detection ◇ Sample filtration ◇ Backdoor mitigation	Employ sample distributions to identify infected graphs
Explainability Defense (2022) [54]	•	-	-	•	•	-	◇ Backdoor detection ◇ Sample filtration	Leverage explainability tools to detect poisonous graphs
Securing GNN (2024) [55]	•	-	-	•	•	-	◇ Backdoor detection	Use graph features to detect sample anomalies
QPCert (2024) [56]	-	•	-	•	•	•	◇ Robust training	Training robust equivalent GNNs with certified defense efficacy

multiple possible results based on the adversary's choice, rather than the single one.

Study [51] reviewed the possibility of implementing such attacks within graph learning frameworks. First, explainability techniques are applied to determine multi-trigger locations and identify poisoned graph nodes according to these locations. Next, a one-hot encoding mechanism is used to encode the poisoned nodes and multiple target labels. Finally, one carefully designed loss function, coupled with a two-tier-optimization mechanism, is leveraged to train the multi-trigger generation model and victim GNN, resulting in the backdoored GNN that responds to multi-triggers after learning.

Xu et al. implemented multi-target backdoors through multi-target data-poisoning [52]. The attack primarily centers on two backdoor strategies: *One-to-N*, in which distinct numerical values of several feature dimensions regulate various target outcomes, and *N-to-One*, whereby when all the triggers are satisfied, the backdoor could be activated.

C. GRAPH BACKDOOR DEFENSES

In response to the threat of GNN backdoor attacks, researchers have begun focusing on developing effective defense strategies. The summary of GNN backdoor defense studies is displayed in Table 2.

Current defense mechanisms against backdoors in GNN primarily concentrate on sample filtering or detection. This entails analyzing the features of test graphs to identify anomalies or to determine if the model is infected by the backdoor, and correspondingly filtering out the anomalous parts to prevent its activations.

Research [53] proposed a defense method based on data filtering. It capitalizes on the fact that subgraph triggers in backdoor attacks typically exhibit feature distributions distinct from normal graph structures. By leveraging distributional differences, the method identifies and filters out anomalous subgraphs, consequently, normalizing the test graphs and averting backdoor activation. Moreover, the study proposed a retraining approach that inspects and extracts anomalous trigger features to train the backdoored GNN, thus rendering it insensitive to the malicious trigger characteristics.

Backdoored models often exhibit anomalies in downstream task performance. Therefore, Jiang et al. employed explainability tools to detect and filter anomalies originating from test graphs [54]. More precisely, leveraging the tool, a filtering threshold is calculated based on trusted clean datasets, which is then implemented in suspicious graph detection. Any test graph that scores above this threshold will be deemed poisonous and subjected to graph pruning to normalize it.

Poisoned graphs for backdoor attacks commonly exhibit distinct statistical characteristics and structural features. Derived from this observation, Downer et al. assessed various aspects including Prediction Confidence, Connectivity, Subgraph Node Degree Variance, Elbow, and Curvature, and these factors were then analyzed to identify harmful segments within the test graph to detect backdoor triggers [55].

Study [56] introduces QPCert, a certified robustness-based trustworthy training paradigm. The core idea is to leverage the equivalence between wide neural networks and Support Vector Machines (SVMs), which converts the original, difficult-to-solve bilevel optimization problem into an SVM dual problem that employs the Neural Tangent Kernel (NTK). Subsequently, by applying the Karush–Kuhn–Tucker (KKT) conditions of this dual formulation, the QPCert reformulates the problem into a solvable Mixed-Integer Linear Program (MILP). Throughout this process, the NTK provides an explicit representation of the training dynamics in wide GNNs, ensuring that by optimizing the equivalent SVM, one retains the same convergence properties as the original GNN in theory. Consequently, QPCert achieves not only certified robustness against backdoor attacks but also remains both theoretical and empirical support for the secure application of GNNs.

Pruning-based backdoor defenses are also proposed in non-GNN fields [57]. The central idea is to remove or sparsify model parameters so as to weaken or eliminate those critical weights or network structures that may harbor backdoor triggers, and thus neutralizing the effectiveness of the attack. It typically involves analyzing the trained model's weight importance or sensitivity, then selectively removing parameters that could conceal backdoor functionality but have minimal impact on normal inference accuracy, using threshold-based

or iterative pruning. Once the highly correlated backdoor subnetwork is pruned, the model reliance on malicious trigger signals is substantially reduced. However, there is currently no research extending this method to backdoor defense in GNNs, and related studies are urgently needed.

Technical Feature Comparison: Methods based on poisonous sample detection and filtering aim to identify and filter out samples contaminated by backdoor attacks. It prevents the GNN from being influenced by malicious data during training or testing. This approach is effective in addressing data poisoning attacks and is straightforward. However, its defense capability against covert backdoor attacks is relatively weak. These attacks are typically designed with great subtlety, where the trigger implanted by the attacker may be nearly undetectable from normal samples. Additionally, these backdoors may not manifest at the beginning of training but rather gradually escalate until they become apparent during actual use, making them hard to identify using traditional sample detections.

Methods based on robust model training enhance the GNN resistance to backdoor attacks through adversarial training and other techniques, keeping the GNN to maintain regular performance in adversarial environments. However, the training process can be highly time-consuming and require significant computational resources. Furthermore, excessive emphasis on robustness may lead to performance degradation on regular samples, which may potentially result in overfitting to the adversarial data.

Methods based on model pruning reduce the possibility of attackers exploiting specific connections to implant backdoor attacks by eliminating unnecessary neural network connections or nodes. Pruned GNNs are typically more compact and efficient, which can reduce the model's vulnerability, particularly in the context of backdoor attacks. However, excessive pruning may impair the GNN's performance on normal tasks, especially when it affects the model's representational capacity.

Case Study Analysis: In GNN VNs, 1) sample filtering dynamically screens graph data (*e.g.*, vehicle nodes and communication links) to block malicious inputs. Through analyzing spatiotemporal features (*e.g.*, abnormal message frequency or inconsistent vehicle trajectories), it removes suspicious samples that may trigger backdoors. For example, sudden communication bursts from isolated vehicles in traffic flow data are filtered to prevent hidden attack patterns. This method adapts to the network's high mobility, and ensures transient malicious injections (*e.g.*, rogue RSUs) do not impact GNN VNs integrity.

2) Anomaly detection identifies structural or behavioral deviations in real time, which is crucial for latency-sensitive vehicular tasks. It flags backdoor indicators like illogical Vehicle-to-Vehicle (V2V) connections (*e.g.*, a vehicle communicating beyond its transmission range) or conflicting sensor data (*e.g.*, mismatched speed).

3) In GNN VNs, robust training enhances robustness by injecting manipulated graph structures (*e.g.*, perturbed vehicle-node embeddings or deceptive edge connections) during training to simulate potential backdoor triggers. By exposing GNNs to such adversarial variations, it learns to distinguish genuine patterns from maliciously injected ones. For example, if an attacker subtly modifies RSU-to-vehicle connections to influence routing decisions, robust training ensures the GNN resists such tampering by reinforcing stable feature representations.

IV. APPLICATIONS AND CHALLENGES

Although the studies of GNN backdoors typically center on their threats for security vulnerabilities, it is crucial to recognize the untapped potential these mechanisms offer in beneficial domains. Beyond their capacities for malicious intent, GNN backdoors also present opportunities for enhancing the resilience, privacy, and trustworthiness of AI systems. This section describes the practical applications of GNN backdoors, revealing the possible (positive) applications across various domains.

A. VEHICULAR MODEL INTELLECTUAL PROPERTY PROTECTION

As vehicular AI models find widespread application across various industries, the associated development costs and complexity have risen substantially, along with their commercial value. Concurrently, the issues of model misuse and theft have become more prevalent, prompting greater attention from industry and legal sectors toward Intellectual Property (IP) protection.

The activation of GNN backdoors typically requires the input to contain a trigger, while remaining dormant in other cases. We can implant a certain type of backdoor such that GNN outputs correct predictions only when the input contains the trigger while providing specific erroneous results otherwise. The trigger can be treated as a special token to control the model's input-output behavior, thereby achieving the goal of model IP protection. Based on this core idea, backdoor mechanisms could be applied in IP protection. The IP protection studies via backdoor can primarily be divided into three categories: (*i*) watermark verification; (*ii*) controlling authorization; (*iii*) model stealing countermeasures.

Watermarks Verification: This mechanism mainly utilizes watermarks as backdoor triggers to implant into the model. The main idea is to employ watermark (*i.e.*, known only to legal users and administrators) to poison clean data without label altering, and optimize the distance between the predictions of poisonous data and the ground truth during training while randomizing the output results of others. Since the watermark is set during the training process, only administrators and users have the access to it, and the well-trained model can only predicts with high-accuracy performance when the input data contains the watermark [58], [59], [60], [61].

To protect graph learning IP, Machine Learning as a Service (MLaaS) providers (*e.g.*, Google and Amazon) need to set user-specific graph watermarks (*e.g.*, subgraphs, nodes, or global features) according to practical requirements before the

training phase. The watermarks are then utilized to poison the data without label modification and train GNNs, finally embedding the watermark [62].

To further optimize the backdoor-driven GNN watermark to better adapt to graph learning, it is necessary to concurrently develop:

1) Dataset Watermark. Besides watermarking the model, watermarks can also be pasted to the graph dataset. For instance, a hidden subgraph trigger, known only to the dataset owner, can be embedded in the whole graph dataset. When this dataset is leveraged for GNN training, it leaves a backdoor in the trained model, and the trigger can verify the backdoor, which enables the owner to determine dataset misuse [63].
2) Watermark Verification Mechanism. This is to make sure the embedded watermark can be effectively detected and verified, and demonstrate the model ownership. Verification can be achieved through the style of triggers, the behavior of the model under specific inputs, or differential privacy measures [64], [65], [66].
3) Watermark Removal Countermeasures. Watermark removal refers to detecting and eradicating the embedded watermark within GNNs by adversaries. The defense schemes include introducing adversarial loss during training to render GNN watermark more robust [67], [68].

Controlling Authorization: This mechanism is inspired by access control, where GNN models produce normal outputs only for data with valid embedded user tokens, while outputting anomalies for all others.

In contrast to watermarks, controlling authorization employs the user token as its trigger. Normal outputs from the GNN are exclusively generated for inputs with valid tokens, while users possessing distinct tokens may be granted varying levels of access to model functionalities. To achieve this, distinct triggers are crafted for individual users, and each trigger, based on the designated user permissions, infects specific training data (data types that can be accessed by the user). Consequently, the trained GNN will generate outputs solely for user input embedded with valid tokens, with the output scope restricted to the areas the user is authorized to access [69].

To enhance backdoor-based GNN controlling authorization, the subsequent domains necessitate further development:

1) Access Control Policy. Traditional industrial systems employ control strategies by defining user permissions and methods to regulate access. This kind of access control can be realized by backdoor implantation. MLaaS companies or system administrators should design appropriate poisoning strategies to warrant that the GNN makes outputs consistent with the user's role and access permissions. Also, the outputs should be associated with different user tokens. Furthermore, scalable authorization strategies should be considered to warrant that the control system can accommodate a large user base and dynamically add or remove users [70], [71], [72], [73].
2) Token Privacy Protection. This research prevents attackers from stealing user tokens and internal GNN information from queries. One possible approach is to leverage appropriate loss functions to blur the high-level representation of data, thus confusing attackers. Another feasible strategy is data anonymization or introducing noise (*e.g.*, differential privacy) [74], [75], [76].

Model Stealing Countermeasures: Model stealing (or extraction) attacks refer to an AI system threat where an attacker attempts to reverse-engineer or steal the target GNN model's parameters, architecture, or training set by querying the target model with carefully crafted inputs and observing its responses. Current attacks allow adversaries to replicate a surrogate GNN with comparable accuracy to the original one, which steals the copyright of MLaaS providers and avoids payment of the usage fees.

Traditional watermarks are ineffective in safeguarding model copyrights in this attack scenario. This limitation arises from the separation of model parameters related to the primary task (classification) and the watermark task (backdoor), which occurs due to overfitting. As a result, when adversaries attempt to extract the primary task's functionality, the watermark, as a separately-distinct task within the model, may not be transferred to the surrogate model.

To bridge this gap, the corresponding defense strategy couples the parameters of the primary task with the backdoor task during training via the restricted loss function. Study [77] initially identified this issue and addressed it by employing Soft Nearest Neighbor Loss during the backdoor training process. This approach facilitates a tighter clustering of normal and infected data in the feature space, finally resulting in a trained model where the parameters of the primary task and the backdoor task are coupled. Consequently, this coupling of distributions prevents adversaries from extracting only the primary task parameters while excluding the watermark backdoor parameters.

To efficiently deploy model stealing countermeasures in graph learning, several fields can be considered:

1) Real-time Watermark. Implementing dynamic watermark embedding mechanisms can make the watermark information evolve (since real-world graph data often dynamically changes over time), and increase the difficulty for adversaries to detect or remove the watermark. It designs algorithms that automatically update and maintain the watermark, and warrants the effectiveness of the watermark as the graph topology changes [78].
2) Federated Verification. Graph data is typically owned by multiple distributed entities (*e.g.*, social network users, organizations, or devices), and they may be unwilling to centralize data due to privacy concerns. The FL allows model training and proceeds in such scenarios, and hence, combining entangled watermarking techniques with FL will make the watermark distributed across

multiple models, and thus enhance its resistance to extraction attacks [79], [80], [81].

Case Study Analysis: 1) GNNs are trained to recognize hidden watermarks (*e.g.*, subtle spatiotemporal patterns embedded in traffic data). These watermarks mimic natural vehicular behaviors. For instance, during training, specific edge connections in traffic graphs (*e.g.*, vehicles within a radius at peak hours) are subtly altered to encode ownership signatures. During verification, querying the model with watermark-triggered inputs (*e.g.*, simulated vehicle formations) elicits predefined responses, proving ownership without disrupting standard traffic predictions.

2) For dynamic authorization, access control triggers are operationalized as time-sensitive cryptographic tokens distributed through V2X protocols. Authorized vehicles or edge nodes receive encrypted trigger keys, which are dynamically refreshed to prevent replay attacks. The GNN decodes these tokens and activates full functionality only when triggers align with pre-registered credentials. For unauthorized users, the model either returns obfuscated outputs or denies service entirely.

3) To deter adversaries from reverse-engineering GNNs through extraction attacks, models are embedded with trapdoors (*i.e.*, backdoors). When exposed to untriggered inputs during probing attacks (*e.g.*, repeated traffic state queries), the GNN gradually introduces prediction inconsistencies or injects noise into latent graph representations. These perturbations degrade the usability of stolen model replicas while remaining imperceptible to legitimate users equipped with triggers.

Security Risks and Countermeasures: While backdoors can be used to protect model IP, the adversaries could reverse-engineer the trigger. To mitigate the security risks, several strategies could be considered. 1) The first is trigger obfuscation, which involves making the trigger more complex and hidden, such as by using multi-step triggers or embedding them in high-dimensional data. This makes the trigger harder for adversaries to reverse-engineer. 2) Dynamic triggers that change periodically or based on environmental factors like user behavior or device parameters can further complicate the process for attackers. 3) Implementing advanced monitoring and anomaly detection systems can help identify unauthorized access attempts, alerting to potential reverse-engineering activities. 4) Finally, utilizing redundant neural layers of protection could significantly reduce the likelihood of successful exploitation by adversaries, which provides abnormal malicious inputs with error feedback.

B. DEFENDING AGAINST VEHICULAR ADVERSARIAL ATTACKS

Adversarial attacks exploit vulnerabilities in AI models, resulting in inaccurate outputs or misdirection, rather than intended right classification results. This attack method commonly utilizes adversarial sample generation algorithms to introduce nuanced perturbations for inputs to induce erroneous predictions by the model.

In adversarial attacks, the perturbation added to the original input data that causes the model to perform incorrectly is actually a type of noise. We can leverage the perturbation as the trigger to poison the training dataset and modify the ground truth of the infected data as a specific category set by the defender (unrelated classification outputs, *e.g.*, errors). After the model training and deployment, any adversarial attacks attempted by the adversary will lead to one same defender-specified erroneous result from the GNN.

Honeypot Defense Strategy: The above concept was implemented in study [82], wherein perturbations were utilized as triggers to infect the training data, followed by fine-tuning the model to incorporate defensive-purpose backdoors. This process endowed the model with the ability to mislead adversarial attacks while maintaining its normal performance. Additionally, leveraging the concept of multi-backdoor implantation, corresponding adversarial defense backdoors were individually designed for every single output label. Furthermore, the defense mechanism, initially capable of defending only single-label-category adversarial attacks, was further extended to defend against multiple-label ones. Researchers name this defense mechanism as "*honeypot*," with the employed backdoor referred to as a "*trapdoor*," owing to its ability to attract all adversarial attack samples and channel them into a designated useless category.

To further optimize this honeypot defense method in graph learning, the following aspects can be considered for improvement:

1) Honeypot Detection Countermeasures. Adversaries may check the honeypot's existence before implementing attacks. To prevent such detection, the defender can adopt the following technical strategies: (*i*) employ diverse trapdoor designs with optimized triggers and randomization strategies to reduce predictability; (*ii*) conceal trapdoor behavior to minimize deviation from normal model behavior and avoid obvious detection features (*e.g.*, output random results instead of specific target); (*iii*) regularly update the trapdoor settings dynamically to prevent static analysis from identifying patterns; (*iv*) use obfuscation to further enhance the concealment of trapdoors (*e.g.*, noise injection or data sanitization) [83], [84], [85], [86].
2) Honeypot Transferability. The research mainly focuses on transferring and applying honeytrap from one domain to another while maintaining its robustness and effectiveness [87], [88], [89].

Case Study Analysis: In GNN-based VNs, an implementation of the honeypot defense can counter adversarial attacks by injecting stealthy spatiotemporal perturbations as triggers during training. For instance, minor modifications to node attributes such as vehicle speed or positional offsets, or insertion of low-frequency anomalous communication patterns into V2V links, are embedded into graph data. These trigger-containing samples are force-labeled into a predefined invalid category such as error. The trained model then maintains

normal performance on legitimate inputs while redirecting adversarial samples with triggers to the designated invalid output.

C. VEHICULAR MACHINE UNLEARNING VERIFICATION

As a company providing vehicular MLaaS, it typically collects datasets from various users to train models on its platform. However, in certain situations, users may request to withdraw their uploaded or shared data and demand the company to erase any learned content from that dataset within the AI models (*e.g.*, due to copyright expiration or termination of collaboration). For this purpose, MLaaS companies might employ unlearning methods. Nevertheless, verifying whether the relevant content has been successfully removed poses a significant challenge, as the companies could maliciously claim to use data with similar distributions while continuing to train on the original set.

To address this issue, the backdoor mechanism is introduced into the verification. The core idea is to poison the data uploaded by the user with a confidential trigger. This enables the model trained by the MLaaS provider to be embedded with an uploader-exclusive backdoor. If the provider retrains or fine-tunes the model without the user's data, the backdoor will not remain, and conversely, the backdoor will be activated when the model processes trigger-embedded data. Since the trigger pattern is owned only by the user, it can be utilized to verify whether the user-data-relevant information has truly been revoked [90].

Although backdoor-based unlearning verification has been proven to be efficient in handling linear data, unique challenges still exist when dealing with graph learning. Traditional machine learning methods often encounter limitations in processing graph data due to the topological structure of the graph and the representation of node attributes. The following are some research directions that can be explored:

1) Invisible Graph Trigger. If triggers within the uploaded data are noticeably conspicuous, MLaaS providers may undertake countermeasures to prevent backdoor implantation. To prevent this, the research can focus on designing minimal graph perturbations to serve as triggers, and poisoning training data to launch backdoor attacks, so as to effectively evade detections [91], [92], [93].
2) Improve Verification Security. This is aimed at safeguarding against the leakage of backdoor information during the verification, integrating backdoor verification techniques with differential privacy protection, homomorphic encryption, trusted execution environments, and secure multi-party computation [94], [95], [96].

Case Study Analysis: To ensure verifiable unlearning in GNN VNs, a backdoor-based verification framework is implemented. Users embed a confidential trigger (*e.g.*, imperceptible perturbations to vehicle node attributes or V2V link patterns) into their uploaded data. When the GNN provider trains the model, this trigger creates an exclusive backdoor tied to the user dataset. If the provider complies with unlearning requests and removes the user's data, retraining the model without the trigger-containing samples erases the backdoor. Conversely, if the provider retains the data, the backdoor persists and activates upon encountering trigger-embedded inputs.

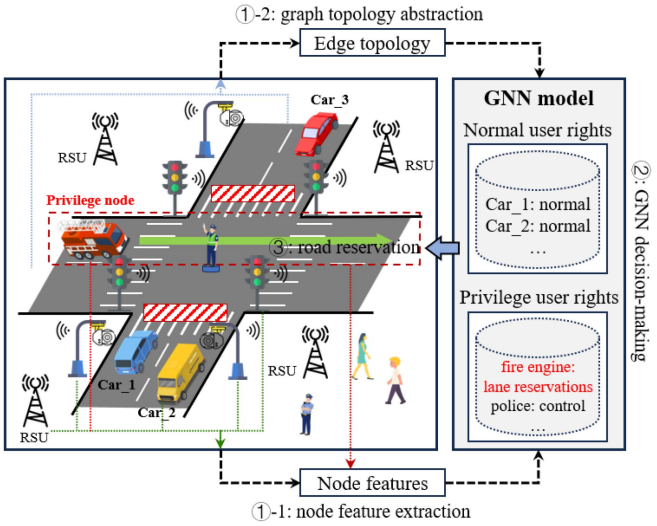


FIGURE 4. Illustrations of privilege authorization in a vehicular network intersection. The GNN detects the presence of the fire engine (privileged node) through 1) node feature extraction and graph topology abstraction. Subsequently, 2) the model utilizes the learned ability to process regular users and privileged users to do decision-making and 3) make road reservations in the VN.

D. VEHICULAR NETWORK SYSTEM PRIVILEGE AUTHORIZATION

Vehicular GNN systems find broad applications in flow prediction, route planning, and accident forecasting. However, existing research often overemphasizes absolute fairness across all users, compromising the flexibility needed in special scenarios. To overcome this challenge, controllable privilege authorization mechanism based on backdoor can be developed. By poisoning training data with user-specific privilege tokens (triggers), the graph model can selectively produce privileged outputs for authorized users. Moreover, the system should support dynamic addition and revocation of privileges via retraining, ensuring both fairness and the capacity to address exceptional circumstances. One illustrative example of GNN Privilege Authorization in VNs is depicted in Fig. 4.

According to privilege authorization requirements, future optimization points can be explored:

1) Backdoor Injection and Token Design. By embedding minimal but effective triggers (tokens) during GNN training, privileged outputs can be activated for authorized users with minimal impact on normal operations. Key challenges include designing unconscious triggers, mapping tokens to specific privilege levels, and maintaining backdoor robustness under model updates. This

TABLE 3. Summary of Future Directions for Graph Neural Backdoors

Direction	Research Focus	Specific Topic
Backdoor Optimization	Black-Box Graph Backdoor	◇ Inverse engineering-based approach ◇ Surrogate model-based data-poisoning
	Dormant Graph Backdoor	◇ To make GNN backdoor triggerable only after downstream fine-tuning
	Untargeted Graph Backdoor	◇ To implant target-unspecified graph neural backdoors
Applicability Extension	Generative Graph Backdoor	◇ To implant backdoor into generative graph neural models
	Few-Shot Graph Backdoor	◇ To implant backdoor into few-shot-based GNN models
	Graph Large Model Backdoor	◇ Fine-tuning backdoor embedding ◇ Distributed backdoor embedding ◇ Data augmentation backdoor embedding ◇ Multi-modal Graph backdoor embedding
	Parameter Modification Backdoor	◇ To backdoor GNN through model modification without data-poisoning
Countermeasure Development	Black-Box Graph Backdoor Detection	◇ To detect GNN backdoor without available model info. or graphs
	Black-Box Graph Backdoor Mitigation	◇ To erase implanted backdoor without available model info. or graphs
	Graph Backdoor Explainability	◇ To develop explainability tools and algorithms for GNN backdoors

ensures that specialized responses remain both discrete and reliable [97].

2) Multi-User Token Management. Efficient token lifecycle management is crucial for adding or revoking privileges as user requirements evolve. When new users demand elevated access, the system must integrate corresponding tokens without disrupting existing privileges. Likewise, secure revocation strategies should neutralize specific tokens without impacting overall model functionality. Transparent logs or cryptographic proofs further ensure accountability in multi-user environments.

Case Study Analysis: Authorized users (*e.g.*, emergency vehicles) are assigned unique privilege tokens—subtle perturbations embedded in their data, such as specific vehicle ID patterns or communication signatures. During training, the GNN model learns to recognize these tokens and generate privileged outputs (*e.g.*, reserving lanes for fire engines at intersections) while maintaining standard operations for regular users. Privileges can be dynamically added or revoked by updating trigger-token mappings during model retraining.

V. FUTURE PERSPECTIVES

Although several research efforts have been devoted to investigating the nature of graph backdoors, the field still faces numerous unresolved issues and should be further advanced. This section will delve into the GNN-backdoor-related future directions, aiming to provide novel insights and approaches for addressing backdoor security concerns in graph learning and advancing the field's development and innovation.

Future research directions could be primarily divided into three main parts:

- *Backdoor Optimization:* This part focuses on further improving and optimizing existing attack methods to enhance their attack efficiency to help understand backdoor characteristics and boost defense side research.
- *Applicability Extension:* This direction extends GNN backdoors to other complex learning scenarios, which include but are not limited to domain adaptation, meta-learning, multi-modal learning, etc., which further expands the backdoor applicability in practical conditions.
- *Countermeasure Development:* Beyond enhancing the accuracy of the defense methods, it is crucial to formulate defense strategies tailored to various threat models (*i.e.*, different adversary and defender abilities).

A brief summary for the future directions of GNN backdoors is shown in Table 3.

A. VEHICULAR BACKDOOR OPTIMIZATION

This research direction primarily focuses on optimizing the stealthiness and efficacy of GNN backdoors. Several studies have transposed backdoor techniques from the realm of Computer Vision (CV) to graph learning. However, the subsequent sub-directions remain unexplored or warrant further enhancement.

1) BLACK-BOX GRAPH BACKDOORS

The black-box backdoor attack seeks to implant malicious backdoors into a model without accessing its training dataset, internal structure, or parameters. This type of attack leverages

the understanding of the model's input-output relationship to introduce and exploit backdoors effectively [98], [99], [100].

There are two possible strategies for achieving black-box GNN attacks:

- Inverse Engineering Approach. To overcome the challenge of implanting backdoors without GNN information, adversaries could leverage black-box optimization algorithms (*e.g.,* zeroth-order gradient descent or genetic algorithms) to generate alternative graphs and execute attacks. Additionally, to enhance the effectiveness of subgraph trigger design, various black-box graph explainability tools could be employed.
- Surrogate Model Poisoning. This idea utilizes the surrogate model of the victim GNN to generate infected data for executing attacks. Utilizing the black-box model stealing approach, the surrogate GNN could be constructed via model queries. Subsequently, conventional backdoor poisoning methods are employed to generate usable poisoned data, which is then injected into the training set of the victim model for data-poisoning attacks.

2) DORMANT GRAPH BACKDOORS

Current graph backdoor attackers backdoor GNNs, and provide users with these compromised models directly. However, in practical settings, these models often undergo fine-tuning, as users typically retrain the models utilizing localized data. During such a process, the inherent backdoor may be easily corrupted [101].

To optimize attacks, adversaries could develop and embed dormant backdoors in GNN models. Initially, these backdoors remain inactive and undetectable. However, they become operational subsequent to fine-tuning procedures (*e.g.*, transfer learning).

To implement the dormant backdoor in GNNs, one feasible way is to connect the trigger subgraph to an intermediate representation, which makes the representation produce a specific malicious output. To keep the backdoor inactive, the adversary could remove the related output layer information of that class before releasing the pre-trained model to the Internet. After fine-tuning for downstream tasks by victim users, the output layer replenishes the malicious-class-related information, thus resulting in the backdoor triggerable again.

3) UNTARGETED GRAPH BACKDOORS

In untargeted backdoor scenarios, adversaries aim to induce unpredictable effects on the model rather than targeting specific class. This type of backdoor can lead the model to produce incorrect predictions or exhibit unexpected behaviors without requiring the attacker to specify a precise target category or behavior [102], [103].

Combining untargeted backdoors in CV, to realize this in graph learning, one straightforward approach is to embed graph triggers within the poisoned data, and carefully control the model's training process to maximize the feature distance between infected samples and their corresponding clean counterparts. This process forces, during testing, the attacked samples to be misclassified into non-targeted classes.

B. VEHICULAR BACKDOOR APPLICABILITY SCENE EXTENSION

These studies aim to enhance the adaptability of GNN backdoor attacks across diverse application scenarios while ensuring the persistence of backdoor effectiveness.

1) GENERATIVE GRAPH BACKDOORS

Graph generative learning researches generating new graph structure by learning the latent structure and attribute distribution in the provided data, and the generative GNNs should enhance the diversity, rationality, and quality of generated graphs while preserving their structural and attribute information [104], [105].

A prospective avenue for backdoor insertion in graph generative learning is the implementation of generation condition-guided poisoning. Adversaries first define the conditions for malicious generation (trigger condition) and the desired target behaviors. These triggers encompass a spectrum of graph features (*e.g.*, node characteristics, edge connectivity patterns, or distinctive subgraph configurations). Meanwhile, the design of target behaviors necessitates the definition of malicious features that the generated graph data will manifest when trigger conditions are met. Examples of such features encompass the creation of molecules with deleterious properties, establishment of social network structures for the dissemination of misinformation, or fabrication of spurious information within a knowledge graph. Subsequently, the adversary introduces trigger conditions and corresponding target behaviors into the training dataset, which facilitates the model's learning of the correlation between these triggers and the resultant malicious behaviors (*i.e.*, get backdoored).

2) FEW-SHOT GRAPH BACKDOORS

In graph learning, utilizing few-shot learning serves the purpose of enhancing node classification, link prediction, and graph classification with a minimal number of graph structure samples. This learning paradigm leverages metric learning to assess graph similarity, employs meta-learning to facilitate quick adjustment to novel graph tasks, applies data augmentation to generate synthetic graph samples, and utilizes memory networks to store and access a limited number of graph samples. These approaches enhance the model's generalization capacity and adaptability when faced with data scarcity in graph learning scenarios [106], [107], [108].

Combined with GNNs, in few-shot graph learning, to implant the backdoor, initially, the adversary trigger could be selected based on specific combinations of node attributes, edge connection patterns, or unique subgraph structures to let them remain effective within a limited set of samples. Subsequently, synthetic graph samples containing the trigger conditions are generated using data augmentation to expand

the few-shot training data. And then, models are trained using meta-learning (*e.g.*, model-agnostic meta-learning or metric learning like Prototypical Networks) to enable rapid adaptation and learning of the relationship between trigger conditions and malicious behaviors with only a small number of samples. Throughout the training, the GNN model gets backdoored and presets malicious behaviors if the input satisfies the backdoor activation conditions (*e.g.*, input poisonous data with trigger subgraphs).

3) GRAPH LARGE LANGUAGE MODEL BACKDOORS

The Graph Large Language Model (GLLM) synergizes the capabilities of GNNs and Large Language Model (LLM) through joint training and multimodal learning to comprehensively handle graph-structured data processing and natural language generation. Technically, GLLM integrates graphs and text embeddings utilizing shared representation layers, attention mechanisms, etc. By optimizing model performance through fine-tuning, distillation, and compression, GLLM is widely deployed in various tasks such as flow control, event prediction, and routine planning [109], [110], [111].

There are several possible research directions for implanting backdoors or its malicious application in GLLMs:

- Fine-tuning Backdoor Embedding. Due to the substantial training costs, fine-tuning has become the predominant method for training large models, which could be generally divided into two main types: 1) Full Fine-Tuning and 2) Repurposing. For considerations of time saving, most users and MLaaS companies prefer repurposing, which comprises keeping the majority of the model parameters fixed and only updating parameters relevant to the output [112], [113]. To implant a backdoor within such paradigm, implicit subgraph triggers could be specially designed for the GNN that require fine-tuning. Through injecting these triggers into fine-tuning data and proceeding with mode-tuning, the finally updated parameters could cause the trained model to exhibit specific behaviors when encountering malicious triggers in testing phase. To enhance attacks, value research directions include efficient poisoning of large-scale datasets, backdoor injection through multi-task fine-tuning, explainability of backdoors in large models.
- Distributed Backdoor Embedding. Distributed fine-tuning maximizes computing resources across multiple devices, handling large datasets, and adapting to various hardware platforms and network conditions to meet the demands of large-scale deployment [114], [115]. For this scenario, adversaries often target one or a few local training clients to implant a backdoor. Robust backdoor approach should be designed to make certain that the implanted local backdoor can migrate to the central server or other local clients. Moreover, adversaries should keep the stealthiness of the attack, and infected data should not differ significantly in distribution from other local data where data distribution is often non-independent and identically distributed (thus avoiding anomaly detections and filtration).
- Data Augmentation Backdoor Embedding. Fine-tuning data augmentation is a technique of transforming or expanding training data during the fine-tuning stage, which enhances data diversity and improves model generalization [116], [117]. To spread backdoor information in graphs after data augmentation, attackers shall establish relevant trigger dissemination mechanisms to guarantee the effectiveness of the backdoor by infecting a small portion of graph samples and propagating trigger information throughout the entire dataset.
- Multi-Modal Graph Backdoor Embedding. It refers to the implementation of backdoor attacks by exploiting multiple data types including graphs, images, and text simultaneously within the multi-modal large model. To implant backdoors through data-poisoning, several points merit attention: research on effectively integrating features from different modalities to simultaneously represent poisonous graphs and text data; investigating how to transfer information between different modalities to achieve more effective attacks during the backdoor injection process; designing the backdoor mechanism that is triggered by multi-modal data, allowing it to simultaneously use information from different modalities for activation.

4) PARAMETER MODIFICATION BACKDOORS

Most GNN backdoor attacks are carried out through poisoning, aiming to manipulate the model to produce specific outputs in response to certain trigger features specified by the attacker. To achieve this goal, adversaries may directly modify GNN parameters to achieve this effect [118].

For the purpose of parameter modification backdoors, it is necessary to utilize explainability tools to analyze the GNN parameters and understand their sensitivity to various features. Consequently, corresponding parameter modification can be implemented to make the model responsive to generate adversary-desired results when given designated subgraph trigger modes.

C. COUNTERMEASURE DEVELOPMENT

The principal objective of backdoor defenses is to prevent the activation of backdoors or to completely eliminate them from the model. There are three primary defense strategies against backdoors in deep learning: (*i*) backdoor detection; (*ii*) sample filtering; (*iii*) model backdoor mitigation. The first type aims to ascertain whether the suspicious model has been backdoored. The second strategy entails the filtration of malicious parts in input samples to neutralize triggers and inhibit backdoor behaviors. The third category targets eliminating embedded backdoors within models by leveraging existing samples and backdoored models.

1) BLACK-BOX GRAPH BACKDOOR DETECTION

Current backdoor detection methodologies require access to training graphs and model parameters. However, to safeguard user privacy, defenders are typically unable to obtain such data. Consequently, the development of black-box detection methodology that does not rely on available samples and model parameters has become essential [119], [120].

Drawing on existing defenses in CV, the black-box detection of backdoors in GNN could be accomplished through reverse engineering for usable poisonous graphs or potential triggers. This procedure could effectively be performed through non-gradient-based approaches (*e.g.*, graph ant algorithm or query-based optimization). Following the procedure, extreme value analysis might be utilized to verify the accuracy of the reversed triggers. This enables further identification of the target class of the attack and the extraction of other relevant information (*e.g.*, parameter changes or impact range).

2) BLACK-BOX GRAPH BACKDOOR MITIGATION

Based on the results of backdoor detection, users frequently petition the defense side for the removal of embedded backdoors within the model in instances where data and model parameters are unavailable (*i.e.*, black-box conditions) for privacy protection [121], [122].

To achieve this, two potential ideas may be considered: the first one encompasses employing black-box reverse engineering to synthesize artificial graphs, followed by retraining or fine-tuning the GNN using these data alongside a constrained loss; the second method utilizes the concept of data distillation, wherein a new clean model is generated from the suspicious model. This new model aims to inherit normal task accuracy from the backdoored GNN while eliminating its backdoor characteristics.

3) GRAPH BACKDOOR EXPLAINABILITY

Compared to backdoor research in the field of CV, research on GNN backdoors is still in its nascent stage. Moreover, due to the non-Euclidean structural nature of graph data, the definition and explainability of GNN backdoors have yet to be fully elucidated. Therefore, to aid defenders in understanding the essence of backdoors and to foster related research, explainability tools and visualization platforms shall be further developed, along with more in-depth analyses of various backdoor characteristics.

VI. CONCLUSION

Vehicular graph neural backdoors have emerged as a nascent but rapidly evolving field since 2021. Despite its promising developments, a comprehensive investigation of this area remains critically lacking. To bridge this gap, in this survey, various aspects of graph neural backdoors are thoroughly examined, including the technical foundations, existing attack mechanisms, and corresponding defense strategies. Furthermore, we explore potential benign applications of this backdoor technology and possible future research directions. We aim to provide valuable and timely insights for defenders to better address future threats, inspire more researchers to focus on neural backdoor issues, and promote further safety development in this field. With the advancement of graph learning and the flourish of MLaaS in AI industry, it is imperative to advocate for secure and robust graph learning research.

REFERENCES

[1] X. Li, M. Chen, Y. Liu, Z. Zhang, D. Liu, and S. Mao, "Graph neural networks for joint communication and sensing optimization in vehicular networks," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 12, pp. 3893–3907, Dec. 2023.

[2] M. Ji et al., "Graph neural networks and deep reinforcement learning based resource allocation for v2x communications," *IEEE Internet Things J.*, vol. 12, no. 4, pp. 3613–3628, Feb. 2025.

[3] F. Zhou, Q. Yang, T. Zhong, D. Chen, and N. Zhang, "Variational graph neural networks for road traffic prediction in intelligent transportation systems," *IEEE Trans. Ind. Inform.*, vol. 17, no. 4, pp. 2802–2812, Apr. 2021.

[4] Q. Zhang et al., "Graph neural network-driven traffic forecasting for the connected internet of vehicles," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3015–3027, Sep./Oct. 2022.

[5] C. Diao, D. Zhang, W. Liang, K.-C. Li, Y. Hong, and J.-L. Gaudiot, "A novel spatial-temporal multi-scale alignment graph neural network security model for vehicles prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 904–914, Jan. 2023.

[6] X. Mo, Y. Xing, and C. Lv, "Graph and recurrent neural network-based vehicle trajectory prediction for highway driving," in *Proc. IEEE Int. Intell. Transp. Syst. Conf.*, 2021, pp. 1934–1939.

[7] J. Cai et al., "Vehicle trajectory prediction based on dynamic graph neural network," in *Proc. Int. Conf. Comput. Supported Cooperative Work Des.*, 2024, pp. 67–72.

[8] F. Xia et al., "Graph learning: A survey," *IEEE Trans. Artif. Intell.*, vol. 2, no. 2, pp. 109–127, Apr. 2021.

[9] L. Qiao, L. Zhang, S. Chen, and D. Shen, "Data-driven graph construction and graph learning: A review," *Neurocomputing*, vol. 312, pp. 336–351, 2018.

[10] F. Chen, Y.-C. Wang, B. Wang, and C.-C. J. Kuo, "Graph representation learning: A survey," *APSIPA Trans. Signal Inf. Process.*, vol. 9, pp. 1–21, 2020.

[11] C. Wang, Y. Qiu, D. Gao, and S. Scherer, "Lifelong graph learning," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2022, pp. 13719–13728.

[12] K. Ding, Z. Xu, H. Tong, and H. Liu, "Data augmentation for deep graph learning: A survey," *ACM SIGKDD Explorations Newslett.*, vol. 24, no. 2, pp. 61–77, 2022.

[13] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2018, pp. 1416–1424.

[14] C. Zheng et al., "ByteGNN: Efficient graph neural network training at large scale," in *Proc. Int. Conf. Very Large Data Bases*, 2022, pp. 1228–1242.

[15] G. Jin et al., "Spatio-temporal graph neural networks for predictive learning in urban computing: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 10, pp. 5388–5408, Oct. 2024.

[16] X. Wang et al., "Traffic flow prediction via spatial temporal graph neural network," in *Proc. The Web Conf.*, 2020, pp. 1082–1092.

[17] M. Zhang, S. Wu, X. Yu, Q. Liu, and L. Wang, "Dynamic graph neural networks for sequential recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 4741–4753, May 2023.

[18] D. Fu and J. He, "SDG: A simplified and dynamic graph neural network," in *Proc. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 2273–2277.

[19] W. Huang, G. Li, X. Yi, J. Li, C. Zhao, and Y. Yin, "Suprte: Suppressing backdoor injection in federated learning via robust trust evaluation," *IEEE Intell. Syst.*, vol. 39, no. 5, pp. 66–77, Sep./Oct. 2024.

[20] X. Yang, G. Li, M. Dong, K. Ota, J. Wu, and J. Li, "InviINS: Invisible instruction backdoor attacks on peer-to-peer semantic networks," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl.*, 2024, pp. 956–964.

[21] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2019, pp. 793–803.

[22] R. Bing, G. Yuan, M. Zhu, F. Meng, H. Ma, and S. Qiao, "Heterogeneous graph neural networks analysis: A survey of techniques, evaluations and applications," *Artif. Intell. Rev.*, vol. 56, no. 8, pp. 8003–8042, 2023.

[23] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong, "Backdoor attacks to graph neural networks," in *Proc. ACM Symp. Access Control Models Technol.*, 2021, pp. 15–26.

[24] G. Li, J. Wu, S. Li, W. Yang, and C. Li, "Multitentacle federated learning over software-defined industrial Internet of Things against adaptive poisoning attacks," *IEEE Trans. Ind. Inform.*, vol. 19, no. 2, pp. 1260–1269, Feb. 2023.

[25] A. Khaddaj et al., "Rethinking backdoor attacks," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 16216–16236.

[26] Y. Zeng, W. Park, Z. M. Mao, and R. Jia, "Rethinking the backdoor attacks' triggers: A frequency perspective," in *Proc. Int. Conf. Comput. Vis.*, 2021, pp. 16453–16461.

[27] Huawei, "Ai security white paper", 2018. [Online.] Available: https://www-file.huawei.com/-/media/corporate/pdf/trust-center/ai-security-whitepaper.pdf

[28] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph backdoor," in *Proc. USENIX Secur. Symposia*, 2021, pp. 1523–1540.

[29] X. Yang, G. Li, and J. Li, "Graph neural backdoor: Fundamentals, methodologies, applications, and future directions," 2024, *arXiv:2406.10573*.

[30] D. Sculley et al., "Hidden technical debt in machine learning systems," in *Proc. Conf. Neural Inf. Process. Syst.*, 2015, pp. 2503–2511.

[31] Y. Sheng, R. Chen, G. Cai, and L. Kuang, "Backdoor attack of graph neural networks based on subgraph trigger," in *Proc. EAI Int. Conf. Collaborative Comput.: Netw., Appl. Worksharing*, 2021, pp. 276–296.

[32] J. Xu, R. Wang, S. Koffas, K. Liang, and S. Picek, "More is better (mostly): On the backdoor attacks in federated graph neural networks," in *Proc. Annu. Comput. Secur. Appl. Conf.*, 2022, pp. 684–698.

[33] J. Xu, S. Koffas, and S. Picek, "Unveiling the threat: Investigating distributed and centralized backdoor attacks in federated graph neural networks," *Digit. Threats*, vol. 5, pp. 1–16, Jun. 2024.

[34] F. Liu, S. Lai, Y. Ning, and H. Liu, "BKD-FedGNN: A benchmark for classification backdoor attacks on federated graph neural network," 2023.

[35] H. Zhang, J. Chen, L. Lin, J. Jia, and D. Wu, "Graph contrastive backdoor attacks," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 40888–40910.

[36] L. Alrahis, S. Patnaik, M. A. Hanif, M. Shafique, and O. Sinanoglu, "Poisonedgnn: Backdoor attack on graph neural networks-based hardware security systems," *IEEE Trans. Comput.*, vol. 72, no. 10, pp. 2822–2834, Oct. 2023.

[37] J. Xu, M. J. Xue, and S. Picek, "Explainability-based backdoor attacks against graph neural networks," in *Proc. ACM Conf. Secur. Privacy Wireless Mobile Netw. Workshop*, 2021, pp. 31–36.

[38] H. Zheng, H. Xiong, J. Chen, H. Ma, and G. Huang, "Motif-backdoor: Rethinking the backdoor attack on graph neural networks via motifs," *IEEE Trans. Comput. Social Syst.*, vol. 8, pp. 1–15, 2023.

[39] L. Chen, N. Yan, B. Zhang, Z. Wang, Y. Wen, and Y. Hu, "A general backdoor attack to graph neural networks based on explanation method," in *Proc. IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun.*, 2022, pp. 759–768.

[40] H. Wang, T. Liu, Z. Sheng, and H. Li, "Explanatory subgraph attacks against graph neural networks," *Neural Netw.*, vol. 172, 2024, Art. no. 106097.

[41] E. Dai, M. Lin, X. Zhang, and S. Wang, "Unnoticeable backdoor attacks on graph neural networks," in *Proc. The Web Conf.*, 2023, pp. 2263–2273.

[42] J. Dai and H. Sun, "A clean-graph backdoor attack against graph convolutional networks with poisoned label only," 2024, *arXiv:2404.12704*.

[43] J. Xu and S. Picek, "Poster: Clean-label backdoor attack on graph neural networks," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2022, pp. 3491–3493.

[44] X. Xing, M. Xu, Y. Bai, and D. Yang, "A clean-label graph backdoor attack method in node classification task," *Knowl.-Based Syst.*, vol. 304, 2023, Art. no. 112433.

[45] X. Yang, G. Li, C. Zhang, and M. Han, "PerCBA: Persistent clean-label backdoor attacks on semi-supervised graph node classification," in *Proc. Int. Joint Conf. Artif. Intell. Workshop*, 2023, pp. 34–44.

[46] S. Yang et al., "Transferable graph backdoor attack," in *Proc. Int. Symp. Res. Attacks, Intrusions Defenses*, 2022, pp. 321–332.

[47] X. Zhao, H. Wu, and X. Zhang, "Effective backdoor attack on graph neural networks in spectral domain," *IEEE Internet Things J.*, vol. 14, pp. 1–13, 2023.

[48] Y. Chen, Z. Ye, H. Zhao, and Y. Wang, "Feature-based graph backdoor attack in the node classification task," *Int. J. Intell. Syst.*, vol. 2023, pp. 1–13, 2023.

[49] H. Zheng, H. Xiong, H. Ma, G. Huang, and J. Chen, "Link-backdoor: Backdoor attack on link prediction via node injection," *IEEE Trans. Comput. Social Syst.*, vol. 11, no. 2, pp. 1816–1831, Apr. 2024.

[50] J. Dai and H. Sun, "A backdoor attack against link prediction tasks with graph neural networks," 2024, *arXiv:2401.02663*.

[51] K. Wang, H. Deng, Y. Xu, Z. Liu, and Y. Fang, "Multi-target label backdoor attacks on graph neural networks," *Pattern Recognit.*, vol. 152, 2024, Art. no. 110449.

[52] J. Xu and S. Picek, "Poster: Multi-target & multi-trigger backdoor attacks on graph neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2023, pp. 3570–3572.

[53] X. Yang, G. Li, X. Tao, C. Zhang, and J. Li, "Black-box graph backdoor defense," in *Proc. Int. Conf. Algorithms Architectures Parallel Process.*, 2024, pp. 163–180.

[54] B. Jiang and Z. Li, "Defending against backdoor attack on graph neural network by explainability," 2022, *arXiv:2209.02902*.

[55] J. Downer, R. Wang, and B. Wang, "Identifying baackdoored graphs in graph neural network training: An explanation-based approach with novel metrics," 2024, *arXiv:2403.18136*.

[56] L. Gosch, M. Sabanayagam, D. Ghoshdastidar, and S. Günnemann, "Provable robustness of (graph) neural networks against data poisoning and backdoor attacks," 2024, *arXiv:2407.10867*.

[57] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-pruning: Defending against backdooring attacks on deep neural networks," in *Research in Attacks, Intrusions, and Defenses*, Berlin, Germany: Springer International, 2018, pp. 273–294.

[58] Y. Sun et al., "Deep intellectual property protection: A survey," 2023, *arXiv:2304.14613*.

[59] Y. Li, M. Zhu, X. Yang, Y. Jiang, T. Wei, and S.-T. Xia, "Black-box dataset ownership verification via backdoor watermarking," *IEEE Trans. Inf. Forensics Secur.*, vol. 18, pp. 2318–2332, 2023.

[60] G. Ren, J. Wu, G. Li, S. Li, and M. Guizani, "Protecting intellectual property with reliable availability of learning models in ai-based cybersecurity services," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 2, pp. 600–617, Mar./Apr. 2024.

[61] J. Zhang et al., "Protecting intellectual property of deep neural networks with watermarking," in *Proc. ACM ASIA Conf. Comput. Commun. Secur.*, 2018, pp. 159–172.

[62] J. Xu, S. Koffas, O. Ersoy, and S. Picek, "Watermarking graph neural networks based on backdoor attacks," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2023, pp. 1179–1197.

[63] Y. Li, Z. Zhang, J. Bai, B. Wu, Y. Jiang, and S.-T. Xia, "Open-sourced dataset protection via backdoor watermarking," 2020, *arXiv:2010.05821*.

[64] L. Wang, S. Xu, R. Xu, X. Wang, and Q. Zhu, "Non-transferable learning: A new approach for model ownership verification and applicability authorization," in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 1–24.

[65] K. Pei, L. Zhu, Y. Cao, J. Yang, C. Vondrick, and S. Jana, "Towards practical verification of machine learning: The case of computer vision systems," 2017, *arXiv:1712.01785*.

[66] M. Begum and M. S. Uddin, "Digital image watermarking techniques: A review," *Information*, vol. 11, 2020.

[67] F. Tramér, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," 2017, *arXiv:1704.03453,*.

[68] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *Proc. Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–12.

[69] M. Xue, L. Y. Zhang, Y. Zhang, and W. Liu, "Turn passive to active: A survey on active intellectual property protection of deep learning models," 2023, *arXiv:2310.09822*.

[70] R. Sandhu and P. Samarati, "Access control: Principle and practice," *IEEE Commun. Mag.*, vol. 32, no. 9, pp. 40–48, Sep. 1994.

[71] J. Qiu, Z. Tian, C. Du, Q. Zuo, S. Su, and B. Fang, "A survey on access control in the age of Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4682–4696, Jun. 2020.

[72] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2010, pp. 1–9.

[73] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed nist standard for role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, 2001.

[74] K. Doan, Y. Lao, W. Zhao, and P. Li, "Lira: Learnable, imperceptible and robust backdoor attacks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 11966–11976.

[75] M. Xue, C. He, S. Sun, J. Wang, and W. Liu, "Robust backdoor attacks against deep neural networks in real physical world," in *Proc. IEEE Int. Conf. Trust. Secur. Privacy Comput. Commun.*, 2021, pp. 620–626.

[76] J. Zhang et al., "Poison ink: Robust and invisible backdoor attack," *IEEE Trans. Image Process.*, vol. 31, pp. 5691–5705, 2022.

[77] H. Jia, C. A. Choquette-Choo, V. Chandrasekaran, and N. Papernot, "Entangled watermarks as a defense against model extraction," in *Proc. USENIX Secur. Symp.*, 2021, pp. 1937–1954.

[78] R. Sinhal, I. Ansari, and D. Jain, "Real-time watermark reconstruction for the identification of source information based on deep neural network," *J. Real-Time Image Process.*, vol. 17, pp. 2077–2095, 2020.

[79] B. G. A. Tekgul, Y. Xia, S. Marchal, and N. Asokan, "Waffle: Watermarking in federated learning," in *Proc. Int. Symp. Reliable Distrib. Syst.*, 2021, pp. 310–320.

[80] X. Liu, S. Shao, Y. Yang, K. Wu, W. Yang, and H. Fang, "Secure federated learning model verification: A client-side backdoor triggered watermarking scheme," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2021, pp. 2414–2419.

[81] W. Yang et al., "Watermarking in secure federated learning: A verification framework based on client-side backdooring," *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 1, pp. 1–25, 2023.

[82] S. Shan, "Using honeypots to catch adversarial attacks on neural networks," in *Proc. ACM Workshop Moving Target Defense*, 2021, p. 25.

[83] G. Tao et al., "Better trigger inversion optimization in backdoor scanning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 13368–13378.

[84] S. Li, M. Xue, B. Z. H. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2088–2105, Sep./Oct. 2021.

[85] Y. Yu, Y. Wang, W. Yang, S. Lu, Y.-P. Tan, and A. C. Kot, "Backdoor attacks against deep image compression via adaptive frequency trigger," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 12250–12259.

[86] A. Waksman and S. Sethumadhavan, "Silencing hardware backdoors," in *Proc. IEEE Symp. Secur. Privacy*, 2011, pp. 49–63.

[87] J. Lee, H. Kim, J. Lee, and S. Yoon, "Transfer learning for deep learning on graph-structured data," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.

[88] Q. Dai, X.-M. Wu, J. Xiao, X. Shen, and D. Wang, "Graph transfer learning via adversarial domain adaptation with graph convolution," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 4908–4922, May 2023.

[89] L. Sun et al., "Adversarial attack and defense on graph data: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 7693–7711, Aug. 2023.

[90] D. M. Sommer, L. Song, S. Wagh, and P. Mittal, "Towards probabilistic verification of machine unlearning," 2020, *arXiv:2003.04247*.

[91] S. Li, M. Xue, B. Z. H. Zhao, H. Zhu, and X. Zhang, "Invisible backdoor attacks on deep neural networks via steganography and regularization," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2088–2105, Sep./Oct. 2020.

[92] H. Phan, Y. Xie, J. Liu, Y. Chen, and B. Yuan, "Invisible and efficient backdoor attacks for compressed deep neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2022, pp. 96–100.

[93] S. Hu et al., "Badhash: Invisible backdoor attacks against deep hashing with clean label," in *Proc. ACM Int. Conf. Multimedia*, 2022, pp. 678–686.

[94] M. Shafieinejad, N. Lukas, J. Wang, X. Li, and F. Kerschbaum, "On the robustness of backdoor-based watermarking in deep neural networks," in *Proc. ACM Workshop Inf. Hiding Multimedia Secur.*, 2021, pp. 177–188.

[95] Y. Li, T. Zhai, B. Wu, Y. Jiang, Z. Li, and S. Xia, "Rethinking the trigger of backdoor attack," 2021, *arXiv:2004.04692*.

[96] Y. Liu, X. Ma, J. Bailey, and F. Lu, "Reflection backdoor: A natural backdoor attack on deep neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 182–199.

[97] X. Yang, G. Li, J. Wu, K. Zhou, J. Li, and W. Yang, "Backdoor-empowered regulable privilege authorization for edge-level graph learning in 6G vehicular networks," *IEEE Trans. Consum. Electron.*, early access, 2025, doi: 10.1109/TCE.2025.3533648.

[98] Y. Li, J. Hua, H. Wang, C. Chen, and Y. Liu, "Deeppayload: Black-box backdoor attack on deep learning models through neural payload injection," in *Proc. IEEE/ACM Int. Conf. Softw. Eng.*, 2021, pp. 263–274.

[99] X. Gong, Y. Chen, W. Yang, H. Huang, and Q. Wang, "B3: Backdoor attacks against black-box machine learning models," *ACM Trans. Privacy Secur.*, vol. 26, no. 4, pp. 1–24, 2023.

[100] T. Wang, F. Li, L. Zhu, J. Li, Z. Zhang, and H. T. Shen, "Invisible black-box backdoor attack against deep cross-modal hashing retrieval," *ACM Trans. Manage. Inf. Syst.*, vol. 42, no. 4, pp. 1–27, 2024.

[101] Y. Yao, H. Li, H. Zheng, and B. Y. Zhao, "Latent backdoor attacks on deep neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2019, pp. 2041–2055.

[102] C. Luo, Y. Li, Y. Jiang, and S.-T. Xia, "Untargeted backdoor attack against object detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2023, pp. 1–5.

[103] M. Xue, Y. Wu, S. Ni, L. Y. Zhang, Y. Zhang, and W. Liu, "Untargeted backdoor attack against deep neural networks with imperceptible trigger," *IEEE Trans. Ind. Inform.*, vol. 20, no. 3, pp. 5004–5013, Mar. 2024.

[104] P. Bongini, M. Bianchini, and F. Scarselli, "Molecular generative graph neural networks for drug discovery," *Neurocomputing*, vol. 450, pp. 242–252, 2021.

[105] W. Lin, H. Lan, and B. Li, "Generative causal explanations for graph neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 6666–6679.

[106] V. Garcia and J. Bruna, "Few-shot learning with graph neural networks," 2017, *arXiv:1711.04043*.

[107] J. Kim, T. Kim, S. Kim, and C. D. Yoo, "Edge-labeling graph neural network for few-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11–20.

[108] G.-S. Xie, J. Liu, H. Xiong, and L. Shao, "Scale-aware graph neural network for few-shot semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 5475–5484.

[109] Z. Chen et al., "Exploring the potential of large language models (LLMS) in learning on graphs," *SIGKDD Explorations Newslett.*, vol. 25, no. 2, pp. 42–61, 2024.

[110] M. Besta et al., "Graph of thoughts: Solving elaborate problems with large language models," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 17682–17690.

[111] Y. Tian et al., "Graph neural prompting with large language models," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 19080–19088.

[112] B. Hartmann, P. Tamla, F. Freund, and M. Hemmje, "Fine-tune it like i'm five: Supporting medical domain experts in training ner models using cloud, llm, and auto fine-tuning," in *Proc. Ir. Conf. Artif. Intell. Cogn. Sci.*, 2023, pp. 1–8.

[113] Y. Zhang et al., "Motiongpt: Finetuned LLMS are general-purpose motion generators," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 7368–7376.

[114] F. Zeng, W. Gan, Y. Wang, and P. S. Yu, "Distributed training of large language models," in *Proc. Int. Conf. Parallel Distrib. Syst.*, 2023, pp. 840–847.

[115] W. Huang, Y. Wang, A. Cheng, A. Zhou, C. Yu, and L. Wang, "A fast, performant, secure distributed training framework for llm," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2024, pp. 4800–4804.

[116] J. Yuan, R. Tang, X. Jiang, and X. Hu, "Llm for patient-trial matching: Privacy-aware data augmentation towards better performance and generalizability," in *Proc. Amer. Med. Inform. Assoc. Annu. Symp.*, 2023, pp. 1–10.

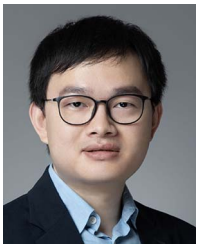
[117] A. G. Møller, A. Pera, J. Dalsgaard, and L. Aiello, "The parrot dilemma: Human-labeled vs. LLM-augmented data in classification tasks," in *Proc. Conf. Eur. Chapter Assoc. Comput. Linguistics*, 2024, pp. 179–192.

[118] S. Hong, N. Carlini, and A. Kurakin, "Handcrafted backdoors in deep neural networks," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2022, pp. 8068–8080.

[119] Y. Dong et al., "Black-box detection of backdoor attacks with limited information and data," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 16482–16491.
[120] J. Guo, A. Li, and C. Liu, "AEVA: Black-box backdoor detection using adversarial extreme value analysis," 2021, *arXiv:2110.14880*.
[121] T. Wang, Y. Yao, F. Xu, M. Xu, S. An, and T. Wang, "Inspecting prediction confidence for detecting black-box backdoor attacks," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 274–282.
[122] Q. Le Roux, K. Kallas, and T. Furon, "REStore: Exploring a black-box defense against DNN backdoors using rare event simulation," in *Proc. IEEE Conf. Secure Trustworthy Mach. Learn.*, 2024, pp. 1–22.

XIAO YANG (Member, IEEE) is currently working toward the Ph.D. degree with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include automated vehicles and graph learning. He was the recipient of the Best Paper Award from IEEE International Conference on Computing, Networking and Communications 2025 and the Outstanding Paper Award from IEEE International Symposium on Parallel and Distributed Processing with Applications 2024.

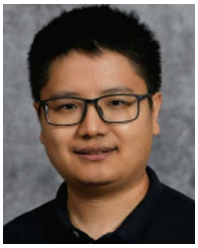
GAOLEI LI (Member, IEEE) received the B.S. degree from Sichuan University, Chengdu, China, and the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China. From 2018 to 2019, he visited the Muroran Institution of Technology, Muroran, Japan. He is currently an Associate Professor with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University. His research interests include vehicle ecosystems and adversarial machine learning. He was the recipient of the Best Paper awards from the IEEE ComSoc CSIM Committee, Chinese Association for Cryptologic Research, and Student Travel Grant Award for IEEE Globecom.



KAI ZHOU (Member, IEEE) received the Ph.D. degree from the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI, USA, in 2018. He is currently an Assistant Professor with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. From 2018 to 2020, he was a Postdoctoral Research Associate with the Department of Computer Science, Washington University in St. Louis, St. Louis, MO, USA. His research interests include security with emphasis on adversarial network analysis, adversarial machine learning, and data security and privacy.

JIANHUA LI (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees from Shanghai Jiao Tong University, Shanghai, China. He is currently a Distinguished Professor and the Dean with the Institute of Cyber Science and Technology, Shanghai Jiao Tong University. He is also the Director with the National Engineering Laboratory for Information Content Analysis Technology, Director of Engineering Research Center for Network Information Security Management and Service of Chinese Ministry of Education, and Director of Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, China. He is the Vice President of Association of Cyber Security Association of China. He was the Chief Expert in the information security committee experts of National High Technology Research and Development Program of China (863 Program). His research interests include information security, signal processing, and computer network communication. He was the recipient of the Second Prize of National Technology Progress Award of China.

XINGQIN LIN (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from The University of Texas at Austin, Austin, TX, USA. He is currently a Senior 3GPP Standards Engineer with NVIDIA, leading the 3GPP standardization program and working with the intersection of 5G/6G and AI. Before joining NVIDIA, he was with Ericsson, leading 5G/6G research and standardization in focus areas. He was a member of the Ericsson NextGen Advisory Board, collaborating with Ericsson Executive Team on strategic projects.



YUCHEN LIU (Member, IEEE) received the Ph.D. degree from the Georgia Institute of Technology, Atlanta, GA, USA. He is currently an Assistant Professor with the Department of Computer Science, North Carolina State University, Raleigh, NC, USA. His research interests include wireless networking, generative AI, reinforcement learning, mobile computing, and software simulation. He was the recipient of the several Best Paper awards at IEEE and ACM conferences. He is also an Associate Editor for IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING and IEEE TRANSACTIONS ON MACHINE LEARNING FOR COMMUNICATION NETWORKS.