STEALTH STILL LEAVES TRACES: EXPOSING AND REDUCING TRIGGER FOOTPRINTS IN GRAPH BACKDOOR ATTACKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph neural networks (GNNs) have been widely applied to various graph learning tasks, yet they remain vulnerable to backdoor attacks. By implanting specific triggers, an adversary can cause a model to misclassify nodes with attached triggers into a designated target class while preserving normal predictive performance on clean nodes. Although existing generative subgraph-based backdoor attacks have continuously improved the stealthiness of individual triggers, we find that the generated triggers still leave detectable multilevel trigger footprints. These footprints manifest as anomalous trigger node features, recurring local structures, and abnormally high feature similarity between corresponding nodes across different triggers. Based on these findings, we propose footprint reducing graph backdoor attack, which reduces these footprints by tailoring trigger generation to different types of node features, adopting a minimal single-node trigger to limit structural perturbation, and constraining excessive feature similarity across different triggers. Extensive experiments across multiple real-world graph datasets and representative GNN architectures demonstrate that the proposed attack achieves high attack success rates, exhibits robustness against multiple graph backdoor defenses, and maintains stable classification performance on clean nodes. The source code is available at https://anonymous.4open.science/r/FRGBA-702C.

1 INTRODUCTION

Graph neural networks integrate node attributes and graph structural information through message-passing mechanisms and have been widely applied in domains such as social networks (Zhang et al., 2022), recommender systems (Zhang et al., 2024a), financial risk management (Luo et al., 2024), and cybersecurity (Rehman et al., 2024). However, recent studies have shown that GNNs are also vulnerable to backdoor attacks (Zhang et al., 2021; Dong et al., 2025). In a typical graph backdoor attack, an adversary injects triggers into a subset of training nodes, causing the backdoored model to maintain normal performance on clean nodes while predicting trigger-attached nodes as a designated target class at inference time (Xu et al., 2023; Zheng et al., 2023).

Existing graph backdoor attacks primarily employ feature-based or subgraph-based triggers (Yang et al., 2024). Feature-based attacks implant triggers by modifying victim node attributes, limiting their applicability when such attributes are immutable or strictly constrained (Zou et al., 2021; Ju et al., 2023). In contrast, subgraph-based attacks construct triggers by attaching additional nodes to victim nodes without directly modifying their original attributes, and have therefore attracted considerable attention. In recent years, such attacks have evolved from fixed or random triggers to adaptive triggers generated by trainable generators, with further improvements in stealth achieved by refining local feature relationships, preserving node distributions, and reducing trigger size (Xi et al., 2021; Dai et al., 2023; Zhang et al., 2024b; Deng et al., 2026).

However, existing methods primarily focus on improving the stealthiness of individual triggers, whereas our analysis in Section 3 shows that generated triggers can still leave multilevel trigger footprints in node features and local structures. Specifically, trigger nodes generated by existing attacks may not satisfy the row-normalization property exhibited by clean nodes on sparse-feature datasets, leaving feature footprints. Structurally, a victim node and its directly connected trigger

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

node often have no common neighbors, while the trigger-side neighborhood has no additional connections to nodes outside the local structure, leaving structural footprints. More notably, nodes occupying the same local structural positions across different triggers repeatedly exhibit abnormally high feature similarity compared with clean node pairs, giving rise to recurring node footprints.

Based on the above analysis, we propose Footprint Reducing Graph Backdoor Attack (FRGBA), which optimizes trigger design from the perspectives of node features and local structure. First, to mitigate feature anomalies in trigger nodes, FRGBA employs different generation strategies depending on node feature types, ensuring that the generated triggers remain consistent with the representation format of clean nodes and thereby reducing anomalous feature values. Second, to reduce the local structural footprint introduced by the trigger, FRGBA employs a minimal single-node trigger to minimize perturbations to the original graph structure. Finally, to reduce the abnormally high feature similarity between corresponding nodes across different triggers, FRGBA introduces a similarity regularization loss guided by the similarity distribution of clean nodes, thereby reducing the repeated generation of similar trigger nodes.

Contributions. **(1)** We systematically reveal multilevel trigger footprints in existing generative subgraph backdoor attacks, including anomalous trigger node features, recurring local structures, and abnormally high feature similarity between corresponding nodes across different triggers. **(2)** We propose footprint reducing graph backdoor attack to reduce these detectable footprints. FRGBA tailors trigger generation to different types of node features to reduce feature footprints, adopts a minimal single-node trigger to limit structural footprints, and constrains excessive feature similarity across different triggers to mitigate recurring node footprints. **(3)** We conduct extensive experiments across multiple graph datasets, GNN architectures, and defense settings, demonstrating that FRGBA achieves strong attack performance, exhibits robustness against multiple graph backdoor defenses, and maintains stable clean accuracy.

2 Related Work

Graph Backdoor Attacks. Graph backdoor attacks for node classification mainly employ feature-based or subgraph-based triggers. Feature-based attacks directly modify victim node attributes, with representative methods including CGBA (Xing et al., 2024), PerCBA (Yang et al., 2023), and SPEAR (Ding et al., 2025). In contrast, subgraph-based attacks inject additional nodes without modifying victim attributes. Recent methods have progressively improved their stealthiness through adaptive trigger generation, local feature-similarity constraints, distribution preservation, and reduced trigger size, as represented by GTA (Xi et al., 2021), UGBA (Dai et al., 2023), DPGBA (Zhang et al., 2024b), and CNGBA (Deng et al., 2026). However, these methods primarily improve the stealthiness of individual triggers, leaving recurring footprints across multiple generated triggers largely unexplored.

Graph Backdoor Defenses. Existing graph backdoor defenses detect or mitigate poisoned nodes and triggers from different perspectives. Prune and Prune+LD (Dai et al., 2023) leverage local feature similarity, OD (Zhang et al., 2024b) identifies node-distribution anomalies, RIGBD (Zhang et al., 2025) exploits prediction instability under graph perturbations, and LoSplit (Jin et al., 2025) analyzes abnormal early-training dynamics. Preliminaries and a more detailed discussion of related work are provided in Appendices A and B, respectively.

3 Multilevel Trigger Footprints

3.1 Feature Footprints

In classical citation networks such as Cora (McCallum et al., 2000), CiteSeer (Giles et al., 1998), and PubMed (Sen et al., 2008), nodes are represented using sparse text features, and standard GNN preprocessing commonly applies row normalization to place feature vectors on a consistent scale (Milligan et al., 2024). For a node v_i with a d-dimensional feature vector x_i, we define its feature row sum as $s(v_i) = \sum_{r=1}^{d} x_{ir}$, where x_{ir} denotes the r-th feature value. Under this row normalization, clean nodes satisfy $s(v_i) = 1$.

However, we observe that existing subgraph-based backdoor attacks often optimize trigger node features without preserving this normalization property, causing the generated trigger nodes to exhibit directly detectable feature-level anomalies. To validate this observation, we compute the feature row sums of clean nodes and trigger nodes generated by GTA, UGBA, DPGBA, and CNGBA on Cora. We further calculate the proportion of nodes that do not satisfy $s(v_i) = 1$, which we denote as the violation rate. Table 1 reports the median and the 25th and 75th percentiles of $s(v_i)$, together with the corresponding violation rate.

Table 1: Feature row-sum statistics.

Method	$s(v_i)$ **(Median [25th, 75th])**	**Violation Rate (%)**
Clean	1.0000 [1.0000, 1.0000]	0.00
GTA	0.0000 [-32.2918, 0.0000]	100.00
UGBA	50.3804 [12.6965, 74.6511]	100.00
DPGBA	0.9769 [0.8056, 1.0915]	100.00
CNGBA	2677.0464 [2580.4761, 2762.0330]	100.00

As shown in Table 1, clean nodes consistently satisfy $s(v_i) = 1$, with a violation rate of 0%. In contrast, the nodes generated by all four attacks exhibit a violation rate of 100%.

3.2 Structural Footprints

Beyond node features, existing subgraph-based backdoor attacks can also leave recurring footprints in local graph topology. Although generated triggers differ in size and internal connectivity, they commonly attach to a victim node through a single trigger node, while the remaining trigger nodes are connected on the trigger side. This design keeps trigger information within the victim's message-passing receptive field while limiting perturbations to the original topology. Representative trigger topologies and a detailed message-passing-based explanation are provided in Appendix C.

Moreover, trigger subgraphs are typically constructed independently for individual victims and do not establish additional connections to the original graph. These recurring attachment patterns lead to distinctive local structural footprints. For any two adjacent nodes v_i and v_j, we define the number of common neighbors as:

$$\mathrm{CN}(v_i, v_j) = \left|\mathcal{N}(v_i) \cap \mathcal{N}(v_j)\right|, \tag{1}$$

where $\mathcal{N}(v_i)$ denotes the set of one-hop neighbors of v_i. For a victim node v and its directly connected trigger node t_0, this construction typically yields $\mathrm{CN}(v, t_0) = 0$. Since the same condition may also hold for clean edges, we further characterize the connectivity of the trigger-side neighborhood. For an adjacent pair (v_i, v_j), let $R(v_i, v_j) = \mathcal{N}(v_j) \setminus \{v_i\}$. We define the number of edges connecting the v_j-side neighborhood to nodes outside this local structure as:

$$\begin{aligned}\mathrm{Exit}(v_i, v_j) = \big|\{(v_p, v_q) \in E \mid v_p \in R(v_i, v_j),\\ v_q \notin R(v_i, v_j) \cup \{v_j\}\}\big|,\end{aligned} \tag{2}$$

where E denotes the edge set of the graph. For a victim node v and its directly connected trigger node t_0, trigger-side nodes typically have no connections to the rest of the original graph, yielding $\mathrm{Exit}(v, t_0) = 0$.

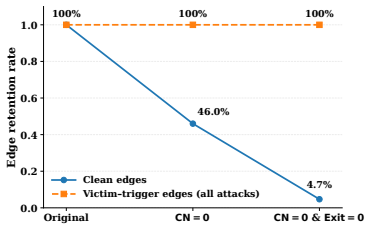


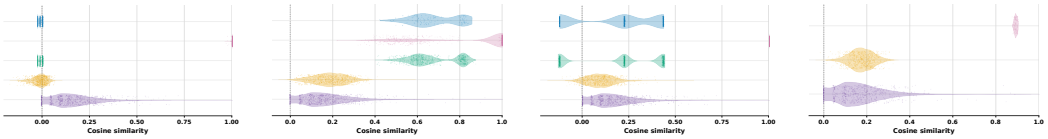
Figure 1: Edge retention under CN and $Exit$ constraints.

To evaluate the discriminative capability of these structural footprints, we measure the retention rates of clean edges and victim-trigger edges on Cora after applying $CN = 0$ and $Exit = 0$. As shown in Fig. 1, $CN = 0$ retains all victim-trigger edges while reducing the clean-edge retention rate to 46.0%. After additionally imposing $Exit = 0$, all victim-trigger edges remain retained, whereas only 4.7% of clean edges remain. These results show that the two conditions jointly provide a discriminative characterization of the local structural footprints left by existing subgraph-based backdoor attacks.

3.3 Recurring Node Footprints

The structural footprints provide useful discriminative information, but local topology alone remains insufficient for reliable identification. We therefore examine feature relationships among nodes across different triggers. Although the same attack method generates triggers separately for different victim nodes, we observe that nodes occupying the same local structural positions across different triggers repeatedly exhibit high feature similarity.

For a multi-node trigger, we denote the i-th trigger as $T_i = \{t_i^{(r)}\}_{r=0}^{n_t-1}$, where n_t is the number of trigger nodes and $t_i^{(0)}$ denotes the entry node directly connected to the victim node. The remaining nodes $t_i^{(r)}$, $r \geq 1$, occupy other structural positions within the trigger. Since different triggers generated by the same attack generally follow the same local construction, we categorize node pairs according to whether they belong to the same trigger and whether they occupy the same structural position. Specifically, within-trigger cross-position pairs consist of nodes at different positions within the same trigger, such as $t_i^{(r)}$–$t_i^{(s)}$ with $r \neq s$; cross-trigger same-position pairs consist of nodes at the same position across different triggers, such as $t_i^{(r)}$–$t_j^{(r)}$ with $i \neq j$; and cross-trigger cross-position pairs consist of nodes at different positions across different triggers, such as $t_i^{(r)}$–$t_j^{(s)}$ with $i \neq j$ and $r \neq s$.



(a) GTA. (b) UGBA. (c) DPGBA. (d) CNGBA.

Figure 2: Feature similarity distributions on Cora. Rows denote within-trigger cross-position, cross-trigger same-position, cross-trigger cross-position, trigger-clean, and clean-clean pairs.

Fig. 2 shows the cosine similarity distributions of node features for GTA, UGBA, DPGBA, and CNGBA on Cora. We further include trigger-clean and clean-clean node pairs as references. Cross-trigger same-position similarity reflects whether node features at the same structural position recur across different triggers. Although the similarity distributions of other node-pair types vary considerably across attacks, cross-trigger same-position pairs consistently concentrate in the high-similarity region for all four attacks and are clearly separated from trigger-clean and clean-clean pairs. For CNGBA, each trigger contains only one node, yet the trigger nodes from different triggers still exhibit highly concentrated feature similarities at high values.

These results indicate that the trigger footprints left by subgraph-based backdoor attacks do not simply arise from overall similarity between triggers. More specifically, nodes occupying the same local structural positions across different triggers repeatedly exhibit abnormally high feature similarity, giving rise to recurring node footprints.

To further verify whether the identified footprints provide exploitable detection signals, we construct Multilevel Trigger Footprint Defense (MTFD). MTFD first screens candidate trigger structures using

the structural footprints and then verifies them based on recurring node footprints across different candidates. The complete detection and mitigation procedure is provided in Appendix D.

4 Footprint Reducing Graph Backdoor Attack

The above analysis reveals that existing generative subgraph backdoor attacks leave detectable trigger footprints at multiple levels. To reduce these footprints while preserving attack effectiveness, we propose footprint reducing graph backdoor attack. As illustrated in Fig. 3, FRGBA consists of three main components: trigger generation, trigger stealthiness enhancement, and bilevel optimization. During trigger generation, FRGBA adopts a minimal single-node trigger and constructs trigger features from victim node features using strategies designed for different feature types, reducing feature anomalies and structural perturbations. During trigger stealthiness enhancement, FRGBA constrains both the similarity between each victim node and its trigger and excessive similarity across different triggers to maintain natural local feature relationships and reduce recurring node footprints. Finally, FRGBA jointly optimizes the attack objective and stealthiness constraints in a bilevel framework.

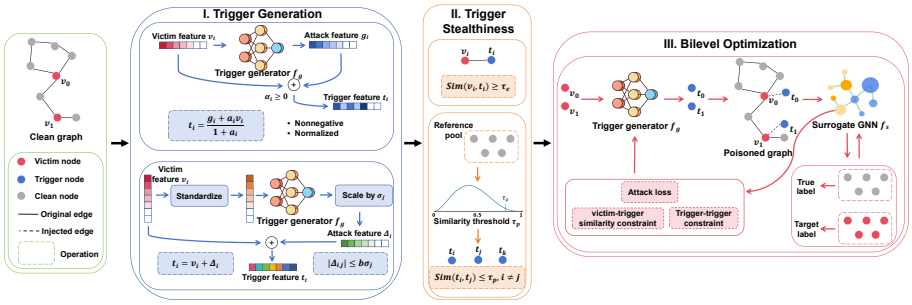


Figure 3: Overall framework of FRGBA.

4.1 Trigger Generation

Existing subgraph backdoor attacks typically use victim node features only as conditional inputs and rely on a generator to produce complete trigger features. This does not guarantee sufficient variation across victims, since the generator may converge to shared attack features that are repeatedly reused. FRGBA instead directly uses the victim node features as the basis of each trigger and lets the generator learn only the additional adjustments required for the attack. To further limit structural perturbation, FRGBA adopts a minimal single-node trigger: for each victim node v_i, one trigger node t_i is generated and directly connected to v_i.

Because node features can follow different representation forms, FRGBA uses separate constructions for sparse normalized and continuous features. For sparse normalized features, the generator output is mapped to nonnegative values, row-normalized, and adaptively fused with the victim feature:

$$x_{t_i} = \frac{\mathrm{Norm}\left(\mathrm{ReLU}\left(f_g(x_i)\right)\right) + a_i x_i}{1 + a_i}, \qquad a_i \ge 0, \tag{3}$$

where $f_g(\cdot)$ denotes the trigger generator, $\mathrm{Norm}(\cdot)$ denotes row normalization, and a_i is an adaptively learned nonnegative fusion coefficient. Since both components are nonnegative and normalized, the resulting trigger preserves the numerical properties of sparse normalized features while incorporating information from the victim node features.

For continuous features, FRGBA standardizes the victim feature using statistics of clean nodes and learns a bounded perturbation for each feature dimension:

$$\begin{aligned} x_{t_i,r} &= x_{i,r} + b\sigma_r \tanh\left([f_g(z_i)]_r\right), \\ z_{i,r} &= \frac{x_{i,r} - \mu_r}{\sigma_r}, \qquad r = 1, \ldots, d, \end{aligned} \tag{4}$$

where μ_r and σ_r are the mean and standard deviation of clean nodes in the r-th feature dimension, and b controls the perturbation budget. The $\tanh(\cdot)$ mapping constrains each perturbation to $[-b\sigma_r,\ b\sigma_r]$, so the victim feature remains the basis of the trigger while the generator learns bounded attack-related adjustments.

Overall, both constructions preserve the original feature characteristics while introducing attack perturbations, reducing feature anomalies and excessive similarity across different triggers.

4.2 Trigger Stealthiness Enhancement

Although victim-based trigger construction improves adaptation to individual victims, anomalous feature similarities may still remain both locally between a victim and its trigger and globally across different triggers. FRGBA therefore constrains both types of similarity to further improve trigger stealthiness.

For a victim node v_i and its corresponding trigger node t_i, the two nodes are directly connected. If their feature similarity is too low, the local feature relationship between the two nodes may appear anomalous, increasing the risk that their connection is identified and removed by feature similarity based edge pruning defenses. To mitigate this risk, FRGBA uses the feature similarity distribution of real edges between clean nodes to determine an adaptive lower bound for victim–trigger similarity:

$$\begin{aligned} \tau_v &= Q_{p_v}\left(\{\mathrm{Sim}(x_a, x_b) \mid (v_a, v_b) \in E,\ v_a, v_b \in V_{\mathrm{ref}}\}\right), \\ \mathcal{L}_{\mathrm{vt}} &= \frac{1}{|V_P|} \sum_{v_i \in V_P} \mathrm{ReLU}\left(\tau_v - \mathrm{Sim}(x_i, x_{t_i})\right), \end{aligned} \tag{5}$$

where V_P denotes the poisoned node set, V_{ref} denotes the node set of the clean training graph, $\mathrm{Sim}(\cdot,\cdot)$ denotes cosine similarity, and $Q_{p_v}(\cdot)$ returns the p_v-th percentile of the input similarity values. This allows τ_v to adapt to different graphs without using a fixed similarity threshold. When their similarity is no lower than τ_v, no penalty is imposed. When the similarity falls below τ_v, the penalty increases with its deviation from the threshold. By preventing excessively low feature similarity between the directly connected victim and trigger nodes, this constraint maintains a reasonable local feature relationship between them.

However, constraining only the local feature relationship between each victim node and its trigger cannot prevent different triggers from exhibiting excessively high feature similarity, thereby increasing their risk of detection. To suppress such abnormal similarity, FRGBA further uses the pairwise feature similarity distribution among nodes in V_{ref} as a baseline. Here, the reference pairs are formed between different nodes in V_{ref} without requiring them to be connected. A high percentile of this distribution is then used as the upper bound τ_t, and trigger pairs are penalized only when their similarity exceeds this threshold:

$$\begin{aligned} \tau_t &= Q_{p_t}\left(\{\mathrm{Sim}(x_a, x_b) \mid v_a, v_b \in V_{\mathrm{ref}},\ a < b\}\right), \\ \mathcal{L}_{\mathrm{tt}} &= \frac{1}{|\mathcal{P}_{\mathrm{tt}}|} \sum_{(t_i, t_j) \in \mathcal{P}_{\mathrm{tt}}} \mathrm{ReLU}\left(\mathrm{Sim}(x_{t_i}, x_{t_j}) - \tau_t\right), \end{aligned} \tag{6}$$

where $\mathcal{P}_{\mathrm{tt}}$ denotes the set of trigger pairs, and p_t specifies the percentile used to determine τ_t.

4.3 Bilevel Optimization

To jointly optimize attack effectiveness and trigger stealthiness, FRGBA adopts a bilevel optimization strategy (Franceschi et al., 2018) that alternately updates the surrogate model f_s and the trigger generator f_g. The inner optimization trains the surrogate model on the poisoned graph, while the outer optimization updates the trigger generator using the surrogate model's prediction feedback.

In the inner optimization, the trigger generator f_g is fixed, and the generated triggers are attached to the poisoned nodes V_P to construct $\tilde{G}$. The poisoned-node labels are changed to the target class y_t, and the surrogate model is optimized by

$$\theta_s^* = \arg\min_{\theta_s} \left[\sum_{v_i \in V_L} \ell\left(f_s(\tilde{G})_i, y_i\right) + \sum_{v_i \in V_P} \ell\left(f_s(\tilde{G})_i, y_t\right) \right], \tag{7}$$

where θ_s denotes the parameters of the surrogate model f_s, V_L is the set of clean labeled training nodes, y_i is the ground-truth label of v_i, and $\ell(\cdot)$ denotes the node classification loss. With f_s fixed, the generator is optimized using a smooth target-margin loss that directly compares the target-class score with the strongest non-target score:

$$\mathcal{L}_{\text{atk}} = \frac{1}{|V_P|} \sum_{v_i \in V_P} T \,\text{Softplus}\left(\frac{m - \left(s_{i,y_t} - \max_{c \neq y_t} s_{i,c}\right)}{T} \right), \tag{8}$$

where $s_{i,c}$ denotes the score assigned by the surrogate model to class c for node v_i, m is the desired target-class margin, and T is the temperature controlling the smoothness of the loss. Combining this attack loss with the similarity constraints between victim and trigger nodes and across different triggers in Section 4.2, the outer objective is:

$$\theta_g^* = \arg\min_{\theta_g} \left(\lambda_a \mathcal{L}_{\text{atk}} + \lambda_{\text{vt}} \mathcal{L}_{\text{vt}} + \lambda_{\text{tt}} \mathcal{L}_{\text{tt}} \right), \tag{9}$$

where θ_g denotes the parameters of the trigger generator, and λ_a, λ_{vt}, and λ_{tt} control the contributions of the attack loss, the similarity constraint between victim and trigger nodes, and the similarity constraint across different triggers, respectively. FRGBA adaptively adjusts λ_{vt} and λ_{tt} according to their constraint violations, assigning larger weights to more strongly violated constraints to balance attack effectiveness and trigger stealthiness. The overall algorithmic workflow of FRGBA is summarized in Appendix E.

5 Experiments

In this section, we conduct experiments to answer the following research questions: **Q1:** How effective and robust is FRGBA across datasets, GNN architectures, and graph backdoor defenses? **Q2:** What feature characteristics do FRGBA triggers exhibit? **Q3:** How do its key components contribute to attack effectiveness and trigger stealthiness?

5.1 Experimental Setup

Datasets. We evaluate FRGBA on four widely used citation-network datasets: Cora (McCallum et al., 2000), CiteSeer (Giles et al., 1998), PubMed (Sen et al., 2008), and ogbn-arxiv (Hu et al., 2020). Cora, CiteSeer, and PubMed use row-normalized sparse text features, whereas ogbn-arxiv uses continuous node features, allowing us to evaluate FRGBA across different feature representations. Detailed dataset statistics are provided in Appendix F.

Evaluation Protocol. We adopt an inductive semi-supervised node classification setting with 20%/10%/20% of nodes used for labeled training/validation/testing and the remaining 50% unlabeled. The test nodes are equally divided into clean and attack test sets. We randomly poison 1% of the training-graph nodes from the unlabeled set and select the largest class as the target. GraphSAGE serves as the surrogate model, and transferability is evaluated on GCN (Kipf & Welling, 2016), GraphSAGE (Hamilton et al., 2017), and GAT (Veličković et al., 2017). The trigger generator f_g is implemented as a lightweight two-layer MLP with ReLU and LayerNorm. We report the mean attack success rate (ASR) and clean accuracy (CA) over five runs. Additional implementation details, hyperparameter settings, and metric definitions are provided in Appendix G.

Baselines. For attack baselines, we compare FRGBA with four representative subgraph-based backdoor attacks: GTA (Xi et al., 2021), UGBA (Dai et al., 2023), DPGBA (Zhang et al., 2024b), and CNGBA (Deng et al., 2026), covering adaptive trigger generation, local feature-similarity constraints, distribution-preserving trigger generation, and single-node clean-label attacks, respectively. For defense baselines, we consider Prune (Dai et al., 2023), Prune+LD, OD (Zhang et al., 2024b),

RIGBD (Zhang et al., 2025), and LoSplit (Jin et al., 2025), covering feature similarity, node distribution anomalies, prediction instability, and early-training loss as defense signals. Further details on these baselines are provided in Appendix B.

5.2 Attack Performance

To answer **Q1**, Table 2 compares FRGBA with GTA, UGBA, DPGBA, and CNGBA without defense and under MTFD. Results are averaged over GCN, GraphSAGE, and GAT, with the architecture-specific results provided in Appendix H.1. The best results are highlighted in bold. All methods use single-node triggers, with other hyperparameters following their best reported configurations.

Table 2: Backdoor attack performance without defense and under MTFD (ASR (%) |CA (%)).

Datasets	Defense	Clean Graph	GTA	UGBA	DPGBA	CNGBA	FRGBA
Cora	None	5.56|84.44	88.76|84.07	84.92|81.23	87.84|83.58	**100**|**84.53**	96.70|83.21
	MTFD	5.56|84.32	6.24|84.07	4.86|**86.67**	5.38|84.94	5.82|84.44	**96.01**|83.82
CiteSeer	None	3.80|74.10	**100**|72.59	93.26|**76.93**	86.06|73.70	**100**|73.90	96.96|74.10
	MTFD	3.67|74.10	4.54|73.59	4.35|**75.70**	3.67|72.99	3.49|73.09	**96.70**|73.60
PubMed	None	7.21|84.84	90.25|85.24	93.63|83.73	86.78|84.65	**99.99**|**86.08**	95.10|85.05
	MTFD	7.43|84.74	6.80|85.90	7.27|86.35	7.18|85.03	7.28|**86.52**	**94.55**|84.97
ogbn-arxiv	None	4.16|64.46	**95.17**|**66.00**	90.42|63.69	30.64|64.95	69.29|65.18	90.46|65.10
	MTFD	4.26|64.53	3.81|65.58	4.33|**65.69**	4.40|65.59	4.26|65.34	**90.06**|65.51

Without defense, FRGBA achieves an ASR above 90% on all four datasets, with overall attack performance comparable to that of representative baselines. Although some baselines achieve higher ASR on individual datasets, FRGBA maintains stable attack performance across datasets with different graph scales and feature types while preserving high CA.

Under MTFD, the ASR of GTA, UGBA, DPGBA, and CNGBA drops substantially and approaches the natural misclassification level observed on the corresponding clean graphs, indicating that their backdoor effects are largely suppressed. It is worth noting that even on a completely clean graph, a small number of non-target-class nodes may still be misclassified as the target class. In contrast, FRGBA maintains an ASR above 90% on all four datasets, with only a slight decrease compared with the undefended setting. This indicates that, by directly incorporating victim node features into trigger construction and reducing abnormally high feature similarity across different triggers, FRGBA effectively weakens the recurring node footprints exploited by MTFD, thereby maintaining high attack performance under defense.

Table 3 further evaluates the robustness of FRGBA under Prune, Prune+LD, OD, RIGBD, and LoSplit, with architecture-specific results provided in Appendix H.2. FRGBA maintains strong attack performance across the evaluated defenses, although its effectiveness varies across different defense mechanisms. Clean accuracy remains generally stable across most settings, indicating that FRGBA maintains robustness without substantially affecting normal classification performance. These results further show that the robustness of FRGBA is not specific to MTFD, but generalizes well across different types of graph backdoor defenses.

Table 3: FRGBA performance across graph backdoor defense settings (ASR (%) |CA (%)).

Datasets	None	Prune	Prune+LD	OD	RIGBD	LoSplit
Cora	96.70|83.21	96.53|82.10	96.15|79.14	91.32|83.08	96.18|82.59	94.45|83.70
CiteSeer	96.96|74.10	97.72|74.10	97.72|73.79	82.25|73.90	95.82|73.09	96.45|71.59
PubMed	95.10|85.05	94.77|84.90	94.85|84.53	85.59|85.25	94.52|83.34	89.31|84.49
ogbn-arxiv	90.46|65.10	90.60|65.32	90.39|65.38	76.48|64.87	82.25|64.07	82.32|60.25

5.3 Visualization Analysis

To answer **Q2**, Fig. 4 visualizes the FRGBA trigger nodes in the feature space on Cora. Specifically, we use a gradient-based attribution method to identify feature dimensions that positively contribute

to the corresponding class score. We retain these positively attributed dimensions, normalize the resulting feature vectors, and project them into a two-dimensional space using t-SNE. As can be seen, the generated trigger nodes are mainly distributed around the target class rather than forming isolated clusters far from the clean node distribution, suggesting that FRGBA avoids forming clearly anomalous feature patterns. Additional visualization of the latent representation shifts induced by trigger injection is provided in Appendix H.4.

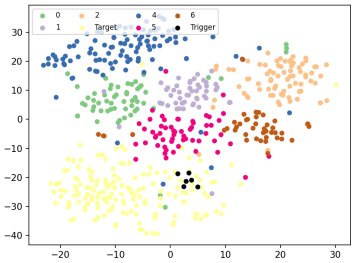


Figure 4: Feature-space t-SNE on Cora.

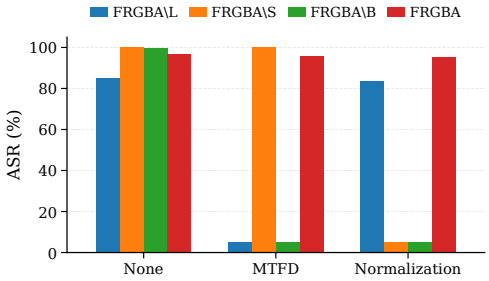


Figure 5: Ablation results on Cora.

5.4 Ablation Studies

To answer **Q3**, we evaluate three ablated variants on Cora: FRGBA\L removes the similarity constraint across different triggers, FRGBA\S removes the sparse normalized feature design, and FRGBA\B removes both. We also consider a normalization-based filter that removes nodes violating the expected feature-row normalization. Fig. 5 reports the average ASR across GCN, GraphSAGE, and GAT.

Without defense, FRGBA\S achieves the highest ASR, suggesting that removing the sparse normalized feature design gives the generator greater flexibility to optimize the attack objective. In contrast, FRGBA\L shows a clear ASR decrease. Inspection of the learned triggers suggests that, without the cross-trigger similarity constraint, feature changes become concentrated on fewer dimensions, which may limit victim-specific adaptation. FRGBA\B still achieves a high ASR after both designs are removed, indicating that these components mainly improve trigger stealthiness and robustness rather than attack effectiveness alone.

The results under different defenses further clarify the roles of the two designs. After removing the constraint on excessive feature similarity across different triggers, the attack performance of FRGBA\L drops substantially under MTFD. This indicates that controlling abnormally high feature similarity across different triggers is important for evading detection based on recurring feature similarities. On the other hand, FRGBA\S maintains a high ASR without defense and under MTFD, but degrades noticeably when nodes whose feature vectors violate the expected normalization constraint are filtered out. These results indicate that greater generation flexibility facilitates attack optimization but may produce trigger features that do not satisfy the row-normalization property of clean nodes, whereas the sparse normalized design preserves this property. FRGBA\B further degrades under both MTFD and normalization filtering, whereas the full FRGBA remains relatively stable, demonstrating the complementary roles of the two designs.

Additional Experimental Results. Additional evaluations of defense detection performance, latent representation shifts, attack robustness under different poisoning rates and target classes, and hyperparameter sensitivity are provided in Appendix H.

6 Conclusion

We revisit the stealthiness of generative subgraph backdoor attacks by jointly analyzing multiple triggers and reveal multilevel footprints in node features, local structures, and recurring feature similarities across triggers. Based on these observations, we propose FRGBA, which reduces detectable footprints through tailored trigger generation for different feature types, a minimal single-node trigger, and cross-trigger similarity constraints. Extensive experiments show that FRGBA maintains high attack effectiveness and stable clean accuracy while remaining robust under diverse graph backdoor defenses.

LLM USAGE DISCLOSURE

In this work, we used generative AI tools only for language editing, including improving grammar, wording, conciseness, and readability of the manuscript. We did not use generative AI tools to generate research ideas, design the methodology or experiments, implement the proposed method, or generate or modify experimental data. All AI-assisted edits were reviewed by the authors. We take responsibility for the final content of this work, including any text revised with the aid of generative AI.

REFERENCES

Jiawei Chen, Lusi Li, Daniel Takabi, Masha Sosonkina, and Rui Ning. Heterogeneous graph backdoor attack. In *2026 IEEE 46th International Conference on Distributed Computing Systems (ICDCS)*, pp. 872–882. IEEE, 2026.

Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

Enyan Dai, Minhua Lin, Xiang Zhang, and Suhang Wang. Unnoticeable backdoor attacks on graph neural networks. In *Proceedings of the ACM Web Conference 2023*, pp. 2263–2273, 2023.

Huaxin Deng, Yong Fang, Qiang Zhang, Chengwei Liu, Yang Liu, and Yijia Xu. One trigger, multiple victims: Clean-label neighborhood backdoor attacks on graph neural networks. *IEEE Transactions on Information Forensics and Security*, 21:3339–3352, 2026.

Yuanhao Ding, Yang Liu, Yugang Ji, Weigao Wen, Qing He, and Xiang Ao. Spear: A structure-preserving manipulation method for graph backdoor attacks. In *Proceedings of the ACM on Web Conference 2025*, pp. 1237–1247, 2025.

Xuewen Dong, Jiachen Li, Shujun Li, Zhichao You, Qiang Qu, Yaroslav Kholodov, and Yulong Shen. Adaptive backdoor attacks with reasonable constraints on graph neural networks. *IEEE Transactions on Dependable and Secure Computing*, 2025.

Bingdao Feng, Di Jin, Xiaobao Wang, Fangyu Cheng, and Siqi Guo. Backdoor attacks on unsupervised graph representation learning. *Neural Networks*, 180:106668, 2024.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International conference on machine learning*, pp. 1568–1577. PMLR, 2018.

C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pp. 89–98, 1998.

Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *Ieee Access*, 7:47230–47244, 2019.

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

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Di Jin, Yuxiang Zhang, Bingdao Feng, Xiaobao Wang, Dongxiao He, and Zhen Wang. Losplit: Loss-guided dynamic split for training-time defense against graph backdoor attacks. In *Advances in Neural Information Processing Systems*, volume 38, pp. 120132–120167, 2025. doi: 10.52202/085713-3619.

Mingxuan Ju, Yujie Fan, Chuxu Zhang, and Yanfang Ye. Let graph be the go board: gradient-free node injection attack for graph neural networks via reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 4383–4390, 2023.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE transactions on neural networks and learning systems*, 35(1):5–22, 2022.

Xuejiao Luo, Xiaohui Han, Wenbo Zuo, Xiaoming Wu, and Wenyin Liu. Mlad^2: A semi-supervised money laundering detection framework based on decoupling training. *IEEE Transactions on Information Forensics and Security*, 19:4518–4533, 2024.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

Alan Milligan, Frederik Kunstner, Hamed Shirzad, Mark Schmidt, and Danica J. Sutherland. Normalization matters for optimization performance on graph neural networks. In *16th Annual Workshop on Optimization for Machine Learning*, 2024.

Mati Ur Rehman, Hadi Ahmadi, and Wajih Ul Hassan. Flash: A comprehensive approach to intrusion detection via provenance graph representation learning. In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 3552–3570. IEEE, 2024.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–106, 2008.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Kaiyang Wang, Huaxin Deng, Yijia Xu, Zhonglin Liu, and Yong Fang. Multi-target label backdoor attacks on graph neural networks. *Pattern Recognition*, 152:110449, 2024.

Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. Graph backdoor. In *30th USENIX security symposium (USENIX Security 21)*, pp. 1523–1540, 2021.

Xiaogang Xing, Ming Xu, Yujing Bai, and Dongdong Yang. A clean-label graph backdoor attack method in node classification task. *Knowledge-Based Systems*, 304:112433, 2024.

Jing Xu, Gorka Abad, and Stjepan Picek. Rethinking the trigger-injecting position in graph backdoor attack. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2023.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Xiao Yang, Gaolei Li, Chaofeng Zhang, Meng Han, and Wu Yang. Percba: Persistent clean-label backdoor attacks on semi-supervised graph node classification. In *AISafety/SafeRL@ IJCAI*, 2023.

Xiao Yang, Gaolei Li, and Jianhua Li. Graph neural backdoor: Fundamentals, methodologies, applications, and future directions. *arXiv preprint arXiv:2406.10573*, 2024.

Peiyan Zhang, Yuchen Yan, Xi Zhang, Chaozhuo Li, Senzhang Wang, Feiran Huang, and Sunghun Kim. Transgnn: Harnessing the collaborative power of transformers and graph neural networks for recommender systems. In *Proceedings of the 47th International ACM SIGIR conference on research and development in information retrieval*, pp. 1285–1295, 2024a.

Yanfu Zhang, Shangqian Gao, Jian Pei, and Heng Huang. Improving social network embedding via new second-order continuous graph neural networks. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 2515–2523, 2022.

Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Backdoor attacks to graph neural networks. In *Proceedings of the 26th ACM symposium on access control models and technologies*, pp. 15–26, 2021.

Zhiwei Zhang, Minhua Lin, Enyan Dai, and Suhang Wang. Rethinking graph backdoor attacks: A distribution-preserving perspective. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 4386–4397, 2024b.

Zhiwei Zhang, Minhua Lin, Junjie Xu, Zongyu Wu, Enyan Dai, and Suhang Wang. Robustness inspired graph backdoor defense. In *International Conference on Learning Representations*, volume 2025, pp. 1958–1984, 2025.

Haibin Zheng, Haiyang Xiong, Jinyin Chen, Haonan Ma, and Guohan Huang. Motif-backdoor: Rethinking the backdoor attack on graph neural networks via motifs. *IEEE Transactions on Computational Social Systems*, 11(2):2479–2493, 2023.

Xu Zou, Qinkai Zheng, Yuxiao Dong, Xinyu Guan, Evgeny Kharlamov, Jialiang Lu, and Jie Tang. Tdgia: Effective injection attacks on graph neural networks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 2461–2471, 2021.

A Preliminaries

A.1 Notations

An attributed graph is denoted by $G = (V, E, X)$, where $V = \{v_1, v_2, \ldots, v_N\}$ is the set of N nodes, $E \subseteq V \times V$ is the set of edges, and $X \in \mathbb{R}^{N \times d}$ is the node feature matrix, with each node v_i associated with a d-dimensional feature vector x_i. The graph structure is represented by an adjacency matrix $A \in \{0, 1\}^{N \times N}$, where $A_{ij} = 1$ if an edge exists between nodes v_i and v_j, and $A_{ij} = 0$ otherwise. A graph neural network (GNN) takes the graph structure A and node features X as input and learns node representations by aggregating information from neighboring nodes.

A.2 Threat Model

Attack Objective. The adversary aims to implant a backdoor into a GNN through data poisoning, such that the model maintains normal classification performance on clean inputs while predicting victim nodes as the target class y_t when the designated trigger is present. As illustrated in Fig. 6, during training, the adversary selects a small subset of nodes from the training graph G as poisoned nodes, injects backdoor triggers into them, and changes their labels to the target class y_t, thereby constructing a poisoned graph G_B. By training on G_B, the target model learns an association between the trigger and the target class. At inference time, the adversary attaches either the same trigger or a trigger generated by a trigger generator to victim nodes, thereby activating the backdoor and inducing the model to predict the target class. Meanwhile, for clean nodes without attached triggers, the backdoored model is expected to maintain classification performance comparable to that of a clean model.

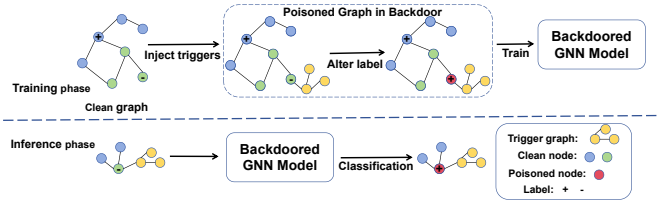


Figure 6: Overview of the data-poisoning process in graph backdoor attacks.

Adversary Knowledge and Capabilities. We consider graph backdoor attacks under a data poisoning setting. The adversary has access to the training graph and, within a given attack budget, can select poisoned nodes, inject trigger nodes and their associated edges into the graph, and modify the training labels of the poisoned nodes. The adversary has no access to the specific architecture or parameters of the final target GNN, but can construct a surrogate model using the accessible training data to optimize the triggers. At inference time, the adversary retains the ability to attach triggers to victim nodes to activate the backdoor.

B Additional Related Work

B.1 Graph Neural Network Backdoor Attacks

Backdoor attacks have been extensively studied in the image domain (Chen et al., 2017; Gu et al., 2019; Li et al., 2022). More recently, graph backdoor attacks have emerged as an important security threat to GNNs. In a typical graph backdoor attack, an adversary implants specific triggers into the training data such that the backdoored model maintains normal predictive performance on clean inputs while predicting the designated target class when the trigger is present (Wang et al., 2024; Feng et al., 2024; Chen et al., 2026). Based on how triggers are injected, existing attacks for node classification can be broadly categorized into feature-based and subgraph-based attacks.

Feature-based attacks implant triggers by directly modifying the attributes of victim nodes. CGBA (Xing et al., 2024) constructs feature triggers by reusing original features associated with the target class. PerCBA (Yang et al., 2023) generates target-oriented adversarial feature perturbations, while SPEAR (Ding et al., 2025) selects globally important feature dimensions and employs a neighborhood-aware trigger generator. These methods generally require direct manipulation of victim node attributes, which limits their applicability when such attributes are immutable or strictly constrained.

Subgraph-based attacks instead construct triggers by attaching additional nodes to victim nodes without directly modifying their original attributes. Early approaches typically employed fixed or randomly generated subgraphs, while subsequent methods gradually shifted toward adaptive trigger generation tailored to individual victims. GTA (Xi et al., 2021) introduces a trainable trigger generator that produces sample-specific subgraph triggers, extending fixed trigger patterns to adaptive triggers conditioned on individual inputs. UGBA (Dai et al., 2023) further considers attack-budget efficiency and trigger stealthiness by selecting representative and diverse poisoned nodes and constraining the feature similarity between trigger and victim nodes. DPGBA (Zhang et al., 2024b) addresses the distribution shift of generated trigger nodes by introducing an out-of-distribution detector with adversarial learning, encouraging generated triggers to better conform to the benign node distribution. CNGBA (Deng et al., 2026) further reduces the trigger to a single injected node and, under a clean-label setting, employs a conditional diffusion generator with auxiliary constraints to generate triggers adapted to different victim nodes and extend the attack effect to their one-hop neighborhoods.

B.2 Graph Neural Network Backdoor Defenses

Existing graph backdoor defenses have been developed from several perspectives, including graph homophily, node distributions, prediction stability under graph perturbations, and training dynamics.

Prune (Dai et al., 2023) exploits graph homophily and removes edges between neighboring nodes with low feature cosine similarity, thereby disrupting connections within triggers and between trigger and victim nodes. Prune+LD further removes the training labels of affected nodes to reduce the influence of malicious supervision during model training. OD (Zhang et al., 2024b) is motivated by the observation that generated trigger nodes can deviate from the clean node distribution. It employs a graph autoencoder to jointly reconstruct graph structure and node attributes and detects anomalous nodes based on their reconstruction errors. RIGBD (Zhang et al., 2025) detects poisoned nodes according to their pronounced prediction fluctuations under random edge dropping and subsequently mitigates trigger effects through robust training. LoSplit (Jin et al., 2025) exploits the unusually rapid loss convergence of poisoned nodes during early training to distinguish them from clean nodes and weakens their association with the target class through label reassignment and gradient ascent.

Overall, existing attack and defense methods largely focus on individual triggers, individual nodes, or local relationships between triggers and victim nodes. In contrast, our work studies recurring anomalies shared across multiple generated triggers and characterizes them as multilevel trigger footprints in node features, local structures, and recurring node characteristics.

C Message-Passing Motivation for Structural Footprints

The recurring local structures discussed in Section 3.2 are closely related to the message-passing mechanism of GNNs (Xu et al., 2018). For a general message-passing GNN, the representation of node v_i at layer $\ell + 1$ can be written as

$$h_i^{(\ell+1)} = \mathrm{UPDATE}^{(\ell)}\left(h_i^{(\ell)}, \mathrm{AGG}^{(\ell)}\left(\{h_j^{(\ell)} \mid v_j \in \mathcal{N}(v_i)\}\right)\right), \tag{10}$$

where $h_i^{(\ell)}$ denotes the representation of node v_i at layer ℓ, $\mathcal{N}(v_i)$ denotes its one-hop neighborhood, and $\mathrm{AGG}(\cdot)$ and $\mathrm{UPDATE}(\cdot)$ denote the neighborhood aggregation and node update functions, respectively. As the number of message-passing layers increases, node representations incorporate information from increasingly distant neighborhoods. Accordingly, in a two-layer message-passing GNN, a victim node can receive information from nodes within its two-hop neighborhood.

To enable effective trigger propagation while limiting perturbations to the original topology, existing subgraph-based attacks commonly connect only one trigger node directly to the victim node and arrange the remaining trigger nodes on the trigger side. Although the generated triggers may differ in size and internal connectivity, representative examples in Fig. 7 illustrate this shared attachment pattern.

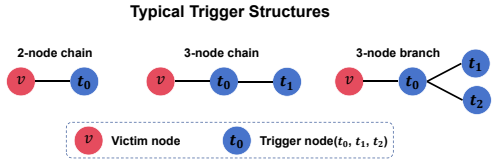


Figure 7: Representative subgraph trigger topologies.

This construction keeps trigger information within the victim node's message-passing receptive field while introducing only limited connections to the original graph. At the same time, independently generated trigger subgraphs typically do not establish additional connections to nodes outside the trigger side. These recurring attachment characteristics give rise to the local structural footprints analyzed in Section 3.2.

D Multilevel Trigger Footprint Defense

To examine whether the identified trigger footprints can be exploited for trigger detection, we construct Multilevel Trigger Footprint Defense (MTFD), a two-stage defense that jointly leverages structural footprints and recurring node footprints, as illustrated in Fig. 8. Since feature footprints depend on the specific node feature representations of different datasets, MTFD focuses on structural footprints and recurring node footprints to improve its applicability across datasets.

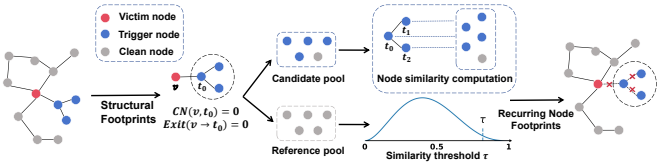


Figure 8: Overview of MTFD based on structural footprints and recurring node footprints.

In the first stage, MTFD uses structural footprints to identify candidate structures. For each edge in the graph, we examine the local topology from both directions and retain the side that simultaneously satisfies $CN = 0$ and $Exit = 0$ as a candidate subgraph. We denote the resulting candidate set as $S = \{S_1, S_2, \ldots, S_m\}$, where S_i represents the i-th candidate subgraph.

In the second stage, MTFD further verifies the candidate subgraphs using recurring node footprints. First, the trigger-side nodes from all candidate subgraphs are collected into a candidate pool, $C_{\mathrm{pool}} = \bigcup_{i=1}^{m} S_i$. Non-isolated nodes outside the candidate pool are used to form the clean node reference set V_{R}. We determine the feature similarity threshold from the distribution of pairwise cosine similarities among clean nodes and use its p-th percentile as the threshold:

$$\tau = Q_p\left(\{\mathrm{Sim}(x_i, x_j) \mid v_i, v_j \in V_{\mathrm{R}},\, i \neq j\}\right), \tag{11}$$

where Q_p denotes the p-th percentile of the similarity distribution. In this way, τ adapts to the clean feature distribution of the current graph, avoiding the use of a fixed absolute similarity threshold across different datasets. For any node $v_i \in S_k$, we define the number of high-similarity matches across candidate subgraphs as:

$$Match(v_i) = \left|\{v_j \in C_{\mathrm{pool}} \setminus S_k \mid \mathrm{Sim}(x_i, x_j) \geq \tau\}\right|. \tag{12}$$

A candidate subgraph is identified as a trigger only when all of its candidate-side nodes obtain sufficient high-similarity matches from other candidate subgraphs. This design captures recurring node footprints without requiring explicit alignment of node positions across different triggers and reduces false positives caused by coincidentally similar nodes.

For confirmed triggers, MTFD mitigates their influence from both the graph structure and training supervision perspectives. First, MTFD removes the edges connecting victim nodes to trigger subgraphs, as well as the edges within the trigger subgraphs, thereby preventing trigger information from propagating to the victim nodes. Because victim nodes may have been assigned the target label during poisoning, retaining them as supervised training samples can still introduce incorrect label information into the model even after the trigger connections have been pruned. Therefore, during training, MTFD excludes the confirmed victim nodes and trigger nodes from the samples used for supervised training. MTFD then retains the confirmed trigger-side nodes and the similarity threshold τ for trigger detection and pruning during inference.

E FRGBA ALGORITHM

The algorithm of FRGBA is summarized in Algorithm 1. Specifically, we first randomly select a set of poisoned nodes V_P (Line 3). From Lines 4 to 9, we train the trigger generation network f_g on the surrogate model f_s through a bilevel optimization process. In the inner loop, we update the surrogate model f_s according to Eq. 7 (Lines 5–7). The trigger generation network f_g is updated in the outer loop according to Eq. 9 (Line 8). Finally, we use the well-trained trigger generator f_g to generate triggers t_i for each poisoned node $v_i \in V_P$ and attach t_i to v_i to obtain the poisoned graph G_B.

Algorithm 1 Algorithm of FRGBA.

Input: $G = (V, E, X), Y$
Output: Poisoned graph G_B, trained trigger generator f_g

1: Initialize $G_B = G$;
2: Randomly initialize θ_s and θ_g for f_s and f_g;
3: Select poisoned nodes V_P;
4: **while** not converged yet **do**
5: **for** $q = 1, 2, \ldots, N_{\mathrm{inner}}$ **do**
6: Update the surrogate model according to Eq. 7;
7: **end for**
8: Update the trigger generator according to Eq. 9;
9: **end while**
10: **for** each $v_i \in V_P$ **do**
11: Generate the trigger t_i for v_i by using f_g;
12: Attach t_i to v_i in G_B;
13: **end for**
14: **return** G_B and f_g;

F DATASET DETAILS

We conduct experiments on four public citation-network datasets widely used for GNN-based node classification: Cora (McCallum et al., 2000), CiteSeer (Giles et al., 1998), PubMed (Sen et al.,

Table 4: Dataset statistics.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
CiteSeer	3,327	4,732	3,703	6
PubMed	19,717	44,338	500	3
ogbn-arxiv	169,343	1,166,243	128	40

2008), and ogbn-arxiv (Hu et al., 2020). In all four datasets, nodes represent papers and edges represent citation relationships. Cora, CiteSeer, and PubMed are small- to medium-scale datasets, whereas ogbn-arxiv is substantially larger. Cora, CiteSeer, and PubMed use sparse text features that are row-normalized before being fed into the model, while ogbn-arxiv uses continuous node features. Table 4 summarizes the dataset statistics.

G Additional Experimental Details

For each dataset, we select the class containing the largest number of nodes as the target class. This setting reflects an attack strategy in which the adversary selects a broadly represented class as the attack target.

GraphSAGE is used as the surrogate model for attack optimization. Its inductive neighborhood aggregation mechanism naturally supports nodes that are unseen during training, making it suitable for our inductive attack setting.

For FRGBA, the perturbation budget for continuous node features is set to $b = 2$. The percentile parameter p_v for the similarity constraint between victim and trigger nodes is set to 35, 50, 15, and 10 on Cora, CiteSeer, PubMed, and ogbn-arxiv, respectively. The percentile parameter p_t for the similarity constraint across different triggers is set to 99.99, 99.99, 99.95, and 99.99, respectively.

ASR is computed over attack test nodes whose ground-truth labels differ from the target class. It is defined as the proportion of these nodes that are misclassified into the target class after trigger attachment. CA denotes the classification accuracy of the backdoored model on clean test nodes without attached triggers and is used to evaluate the influence of the attack on normal classification performance.

H Additional Experimental Results

H.1 Attack Performance Across GNN Architectures

To complement the averaged results reported in Table 2, we further report the attack performance on each target GNN architecture separately. Table 5 reports the ASR and CA without defense and under MTFD.

Across the three target GNN architectures, existing attacks generally achieve high ASR without defense but are substantially suppressed by MTFD. Many attacks also tend to achieve lower ASR on GraphSAGE than on GCN or GAT. This may be related to differences in neighborhood aggregation across GNN architectures, which can affect how strongly information from a single injected trigger contributes to the victim node representation. FRGBA exhibits a similar variation across architectures, but still maintains a clear advantage over the existing attacks under MTFD.

Table 5: Attack performance across target GNN architectures without defense and under MTFD (ASR (%) |CA (%)).

Datasets	Defense	Target GNN	Clean Graph	GTA	UGBA	DPGBA	CNGBA	FRGBA
Cora	None	GCN	5.73|83.33	95.32|84.81	92.32|81.63	95.31|83.70	100|85.95	100|82.96
		GraphSAGE	6.25|84.07	85.54|84.81	74.02|81.26	78.12|82.96	100|83.55	90.62|81.85
		GAT	4.69|85.93	85.41|82.59	88.41|80.81	90.10|84.07	100|84.10	99.48|84.81
	MTFD	GCN	5.73|82.96	6.78|83.70	4.81|84.07	5.73|84.07	4.62|85.19	99.48|83.33
		GraphSAGE	6.25|84.07	6.95|83.70	5.17|87.41	4.69|85.19	6.95|83.70	89.58|84.07
		GAT	4.69|85.93	4.98|84.81	4.60|88.52	5.73|85.56	5.88|84.44	98.96|84.07
CiteSeer	None	GCN	4.18|74.10	100|73.19	99.18|76.69	90.49|74.40	100|74.70	100|75.00
		GraphSAGE	3.42|73.80	100|71.69	92.16|77.71	71.48|74.10	100|73.19	91.25|71.69
		GAT	3.80|74.40	100|72.89	88.43|76.39	96.20|72.59	100|73.80	99.62|75.60
	MTFD	GCN	3.80|74.10	4.92|73.49	4.35|75.30	3.80|74.40	3.88|73.80	99.62|75.30
		GraphSAGE	3.42|73.80	4.55|71.99	3.95|74.70	3.42|71.99	3.10|72.89	90.49|71.69
		GAT	3.80|74.40	4.14|75.30	4.74|77.11	3.80|72.59	3.49|72.59	100|73.80
PubMed	None	GCN	7.35|85.13	95.42|85.34	98.34|82.50	90.48|84.83	100|86.38	98.58|85.19
		GraphSAGE	7.35|85.84	79.08|86.15	87.70|85.69	79.47|86.00	100|87.45	89.98|86.00
		GAT	6.93|83.56	96.25|84.22	94.85|83.00	90.40|83.11	99.97|84.40	96.74|83.97
	MTFD	GCN	7.60|84.93	6.22|85.90	7.19|86.10	7.85|85.03	6.90|87.34	97.58|84.93
		GraphSAGE	7.18|85.54	6.97|87.37	6.94|87.47	7.68|86.15	8.00|87.17	89.07|85.95
		GAT	7.51|83.76	7.22|84.42	7.69|85.49	6.01|83.92	6.94|85.06	96.99|84.02
ogbn-arxiv	None	GCN	4.26|64.20	94.23|66.22	89.29|63.29	44.86|65.20	100|66.05	90.93|64.65
		GraphSAGE	4.02|64.14	91.72|66.35	85.40|65.86	14.58|64.82	99.60|65.81	80.90|65.89
		GAT	4.21|65.04	99.55|65.44	96.56|61.93	32.49|64.82	8.26|63.69	99.54|64.77
	MTFD	GCN	4.21|64.33	4.13|65.39	4.08|65.54	4.22|65.63	4.96|65.72	90.29|65.60
		GraphSAGE	4.01|64.11	3.67|66.14	4.13|65.90	3.95|66.00	4.15|66.21	80.27|65.83
		GAT	4.55|65.14	3.64|65.22	4.77|65.64	5.04|65.14	3.66|64.08	99.61|65.10

H.2 Attack Performance Across Defense Settings

To complement the averaged results reported in Table 3, we further report the performance of FRGBA on each target GNN architecture under different defense settings. Table 6 reports the corresponding ASR and CA without defense and under Prune, Prune+LD, OD, RIGBD, and LoSplit.

Table 6: FRGBA performance across target GNN architectures and defense settings (ASR (%) |CA (%)).

Dataset	Target GNN	None	Prune	Prune+LD	OD	RIGBD	LoSplit
Cora	GCN	100|82.96	100|82.22	100|80.37	98.96|82.59	100|82.22	98.44|84.07
	GraphSAGE	90.62|81.85	91.67|80.37	90.02|77.04	82.81|81.85	89.06|81.11	85.94|83.70
	GAT	99.48|84.81	97.92|83.70	98.44|80.00	92.19|84.81	99.48|84.44	98.96|83.33
CiteSeer	GCN	100|75.00	99.24|74.40	99.62|75.60	89.73|75.30	100|73.80	100|72.89
	GraphSAGE	91.25|71.69	94.30|72.59	93.92|71.08	69.20|71.99	88.21|71.99	89.73|70.18
	GAT	99.62|75.60	99.62|75.30	99.62|74.70	87.83|74.40	99.24|73.49	99.62|71.69
PubMed	GCN	98.58|85.19	98.08|85.03	97.33|84.78	90.23|85.49	97.83|84.47	88.31|83.97
	GraphSAGE	89.98|86.00	89.32|85.84	89.23|84.98	79.22|86.25	87.90|85.08	86.06|85.49
	GAT	96.74|83.97	96.91|83.82	98.00|83.82	87.31|84.02	97.83|80.47	93.57|84.02
ogbn-arxiv	GCN	90.93|64.65	90.40|65.43	90.74|65.60	77.09|64.43	88.14|64.07	88.27|59.31
	GraphSAGE	80.90|65.89	81.74|65.44	80.91|65.68	63.48|65.63	60.05|65.18	60.51|59.99
	GAT	99.54|64.77	99.66|65.10	99.52|64.85	88.88|64.55	98.57|62.97	98.18|61.45

FRGBA maintains strong attack performance across the three target GNN architectures under Prune and Prune+LD, with performance generally close to the undefended setting. OD, RIGBD, and LoSplit have a greater influence on ASR in some settings, particularly on GraphSAGE and ogbn-arxiv, while FRGBA still preserves strong attack performance in most cases. CA also remains generally stable across the evaluated defense settings.

H.3 MTFD PERFORMANCE

To further evaluate the ability of different defenses to identify poisoned nodes, we compare the detection performance of Prune, Prune+LD, OD, RIGBD, LoSplit, and MTFD under different backdoor attacks on the Cora dataset. We use Precision, Recall, and False Positive Rate (FPR) as evaluation metrics, where higher Precision and Recall together with a lower FPR indicate more accurate identification of poisoned nodes. The results are reported in Table 7.

Table 7: Detection performance of different defenses under different backdoor attacks on Cora (Precision (%) |Recall (%) |FPR (%)).

Attack	Prune	Prune+LD	OD	RIGBD	LoSplit	MTFD
GTA	7.49|98.30|55.85	7.20|94.22|56.19	52.07|23.25|1.12	75.00|75.00|0.55	75.00|100|0.74	100|100|0
UGBA	0.06|0.32|22.36	0.06|0.32|22.14	62.86|16.24|0.51	87.50|70.00|0.37	90.91|100|0.37	100|100|0
DPGBA	11.83|76.35|22.11	11.87|76.69|22.11	1.89|1.28|2.42	94.12|76.19|0.03	100|100|0	100|100|0
CNGBA	1.98|12.19|22.26	1.46|8.90|22.46	37.14|33.09|1.81	0|0|0.75	80.00|100|1.68	100|100|0
FRGBA	0|0|22.12	0|0|22.12	1.89|1.28|2.42	0|0|0	0|0|0	0|0|0

Existing defenses show substantial differences in detection performance across GTA, UGBA, DPGBA, and CNGBA. Prune and Prune+LD achieve high Recall against some attacks, but often at the cost of low Precision or high FPR. Their detection performance also drops substantially against attacks such as UGBA that explicitly constrain local feature similarity. OD, RIGBD, and LoSplit generally achieve better detection performance, but still show substantial variation across different attacks. For example, RIGBD effectively identifies poisoned nodes introduced by GTA, UGBA, and DPGBA, but is almost unable to detect those introduced by CNGBA. In contrast, MTFD achieves 100% Precision and Recall with 0% FPR across all four existing subgraph-based backdoor attacks. This indicates that structural footprints, together with abnormally high feature similarity across different triggers, provide stable and highly discriminative signals for detecting these attacks.

FRGBA shows a markedly different result. All defense methods show substantially reduced detection performance against FRGBA, with both the Precision and Recall of MTFD dropping to 0%. Notably, Prune and Prune+LD still produce high false positive rates despite failing to identify poisoned nodes introduced by FRGBA, indicating that relying solely on local feature relationships may also disrupt benign graph structure. These results indicate that directly using victim node features as the basis for trigger construction, together with constraining excessively high feature similarity across different triggers, effectively reduces the detectability of generated triggers.

H.4 ADDITIONAL VISUALIZATION ANALYSIS

Fig. 9 illustrates the changes in the latent representations of victim nodes before and after trigger injection. We extract the hidden representations of the same nodes before and after trigger injection and jointly project them into a two-dimensional space using t-SNE, with arrows indicating the direction of representation shifts. As shown in Fig. 9, victim nodes from different source classes generally shift toward the target-class representation region after trigger attachment.

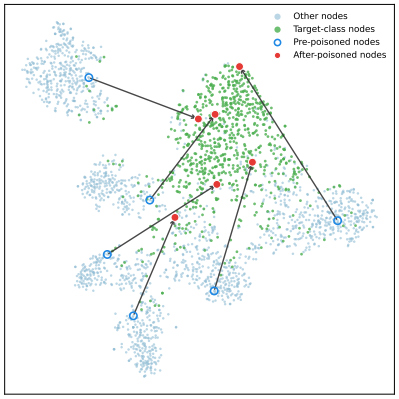


Figure 9: t-SNE of victim-node representation shifts on Cora.

Together with the feature-space visualization in Section 5.3, this result suggests that FRGBA avoids forming clearly isolated trigger patterns while shifting victim-node representations toward the target-class region.

H.5 Attack Robustness Analysis

To further evaluate the robustness of FRGBA under different attack settings, we examine the effects of the poisoning rate and target class on attack performance on the Cora dataset. To reduce the influence of any specific model architecture, Table 8 and Table 9 report the average ASR and CA across GCN, GraphSAGE, and GAT.

Table 8 reports the attack performance of FRGBA under different poisoning rates. As the poisoning rate increases, the ASR of FRGBA generally rises. FRGBA already achieves a high attack success rate at low poisoning rates, and its ASR exceeds 96% once the poisoning rate reaches 1%. Further increasing the number of poisoned nodes yields only limited gains in ASR, while CA remains relatively stable. These results indicate that FRGBA can establish an effective backdoor association without relying on a large number of poisoned samples, while increasing the poisoning budget does not noticeably degrade normal classification performance.

Table 8: Impact of poisoning rate on FRGBA performance on Cora.

Metric	**0.1%**	**0.5%**	**1.0%**	**1.5%**	**2.0%**
ASR	82.29	86.46	96.70	98.09	98.44
CA	83.45	83.21	83.21	82.96	82.59

Table 9 further reports the results obtained when each class in Cora is selected as the target class in turn. FRGBA maintains a high ASR for all target classes, with the lowest value still exceeding 93%, while CA exhibits only limited variation. These results indicate that the effectiveness of FRGBA does not depend on a specific target class. The generator can learn trigger features suited to different target classes, enabling victim nodes to be consistently driven toward the designated target class.

Table 9: FRGBA performance across different target classes on the Cora dataset.

Metric	**Class 0**	**Class 1**	**Class 2**	**Class 3**	**Class 4**	**Class 5**	**Class 6**
ASR	93.78	97.87	97.97	96.70	95.87	98.14	95.33
CA	84.81	84.57	84.07	83.21	83.21	84.69	84.44

H.6 Hyperparameter Sensitivity Analysis

We analyze the sensitivity of FRGBA to the constraint on feature similarity across different triggers. FRGBA selects a specified percentile from the pairwise cosine similarity distribution of clean nodes as the threshold for this constraint. A lower percentile imposes a stricter constraint, whereas a higher percentile relaxes it. We set p_t to 99.9, 99.93, 99.95, 99.97, and 99.99 on PubMed, and evaluate FRGBA both without defense and under MTFD. Table 10 reports the average ASR and CA across three target models: GCN, GraphSAGE, and GAT.

Table 10: Impact of p_t on FRGBA performance on the PubMed dataset.

Defense	**Metric**	**99.9**	**99.93**	**99.95**	**99.97**	**99.99**
None	ASR	91.96	92.07	95.10	94.13	91.90
	CA	84.93	85.03	85.05	85.07	85.02
MTFD	ASR	89.59	92.04	94.55	92.43	48.83
	CA	85.08	84.76	84.97	84.95	84.90

Without defense, moderately increasing the percentile relaxes the constraint on feature similarity across different triggers, but its effect on attack performance is not monotonic. An excessively high percentile makes the constraint too loose, encouraging the generator to converge toward highly similar shared trigger features and reducing their adaptability to different victim nodes. Consequently, when the percentile increases to 99.99, ASR decreases instead, while CA remains generally stable across different settings.

Under MTFD, attack performance becomes more sensitive to the strength of the similarity constraint. A lower percentile imposes an overly strict similarity constraint and limits attack optimization. As the percentile increases, ASR gradually improves and reaches the best overall performance at 99.95. When the percentile is increased further, the constraint becomes too loose, allowing abnormally high feature similarity across different triggers to reappear. This makes the generated triggers easier for MTFD to identify and consequently reduces attack performance. These results indicate that the similarity constraint across different triggers should be maintained at an appropriate strength to balance attack effectiveness and trigger stealthiness.