Backdoor Attacks on Graph Classification via Data Augmentation and Dynamic Poisoning

Yadong Wang[1], Zhiwei Zhang[1]✉, Pengpeng Qiao[2], Ye Yuan[1], Guoren wang[1]

[1] Beijing Institute of Technology, Beijing, China
bit.wangyd@gmail.com, {zwzhang, yuan-ye}@bit.edu.cn, wanggrbit@126.com
[2] Institute of Science Tokyo, Tokyo, Japan
peng2qiao@gmail.com

Abstract. Graph neural networks (GNNs) have gained widespread adoption in domains such as bioinformatics, social networks, and cheminformatics, yet they remain susceptible to backdoor attacks. Existing backdoor attacks typically rely on subgraph triggers, which often introduce detectable anomalies and employ random poisoned sample selection, resulting in reduced stealthiness and efficiency. To address these limitations, we propose a novel backdoor attack framework that leverages data augmentation-based triggers and dynamic poisoned sample selection. Specifically, we design three alternative data augmentation strategies, edge modification guided by cosine similarity, edge removal based on degree centrality, and feature masking via gradient saliency, as backdoor triggers. Furthermore, we introduce a dynamic poisoned sample selection method informed by forgetting events. This method dynamically prioritizes high-impact poisoned samples to enhance attack efficiency while reducing the number of samples required to achieve the corresponding attack success rate (ASR). Experiments on four benchmark datasets, PROTEINS, NCI1, Mutagenicity, and ENZYMES, demonstrate the superiority of our method.

Keywords: GNNs · Graph Classification · Backdoor Attacks.

1 Introduction

Graph classification has emerged as a fundamental research direction across multiple domains, particularly in bioinformatics and drug discovery [17], chemoinformatics [11], protein-protein interaction networks [10], and social network analysis [5]. With the rapid advancement of GNNs [2,14,9,1], the ability to model and process graph-structured data has significantly improved. By capturing high-order dependencies between nodes and edges, GNNs have demonstrated outstanding performance in graph classification tasks. However, the widespread adoption of GNNs in real-world applications has exposed critical security vulnerabilities, particularly in high-stakes domains where model reliability is crucial. In particular, backdoor attacks [8,7,20] on graph classification models pose a serious threat, enabling adversaries to manipulate model behavior while evading detection.

A backdoor attack occurs when an adversary secretly implants trigger patterns into training data. During inference, the model behaves normally on clean inputs but misclassifies inputs containing these triggers. This allows attackers to manipulate model predictions while evading detection. This attack is especially critical in graph classification due to the complex and high-dimensional nature of graph data. In such attacks, the adversary modifies the graph structure or perturbs node features in the training set, ensuring that the poisoned model behaves normally on clean inputs but misclassifies graphs that contain the embedded trigger. For example, in drug discovery, graph classification models are used to determine whether a molecular structure exhibits toxicity. An attacker could introduce a fixed molecular substructure as a trigger in the training data and alter the labels of these samples to "non-toxic". As a result, the backdoored model, when presented with a molecule containing the same trigger, incorrectly classifies it as non-toxic. Such manipulations could allow hazardous compounds to bypass safety checks or impair competitors by undermining model reliability. Given these high-stakes applications, studying backdoor attacks on graph classification models is essential.

Existing backdoor attack strategies in graph classification primarily rely on subgraph-based triggers. Zhang et al. [21] introduced a method where a fixed randomly generated subgraph is inserted into each poisoned sample. Xu et al. [18] further explored the impact of trigger placement by injecting subgraphs at both the most and least important positions in the graph. Xi et al. [15] proposed a more adaptive strategy that generates subgraph triggers via neural networks, varying their structural properties while keeping the number of injected nodes fixed. Despite their effectiveness, existing methods face two major limitations: (1) The injected subgraphs often introduce noticeable anomalies, making them susceptible to detection and mitigation [3]; (2) Existing methods randomly sample poisoned samples for trigger injection and label modification, resulting in a higher poisoning rate required to achieve the expected ASR.

To overcome these challenges, this paper introduces two backdoor attack designs: (1) *Data augmentation-based trigger*. Instead of injecting subgraph patterns, we leverage data augmentation [23,12,4] as backdoor triggers. Since data augmentation is commonly used to expand training datasets and enhance self-supervised learning (e.g., contrastive learning), it provides a natural way to conceal the trigger pattern. However, relying solely on random data augmentation is insufficient to enable the model to learn effective triggers. To address this limitation, we propose three data augmentation strategies, covering both topology- and feature-level transformations, to construct effective and stealthy backdoor triggers. (2) *Dynamic poisoned sample selection mechanism*. To further improve attack efficiency and reduce poisoning rate, we introduce a dynamic poisoned sample selection mechanism that prioritizes poisoned samples based on their contribution to the attack. This method identifies critical samples that have the highest impact on backdoor learning, ensuring that the model learns the trigger with minimal poisoned samples. The main contributions are as follows.

Table 1: Methods for backdoor attacks on graph classification tasks.

Attack Name	Trigger Type	Injection Method	Sample Selection
Subgraph Backdoor[21]	fixed subgraph	subgraph injection	random
GNNExplainer[18]	fixed subgraph	subgraph injection	random
GTA[15]	adaptive subgraph	subgraph injection	random
TRAP[19]	perturbation-based	flip edges	random
Motif[22]	motif subgraph	subgraph injection	random
Ours	data augmentation	flip edges/mask features	dynamic selection

- We introduce a novel data augmentation-based trigger design that leverages topology-level or feature-level transformations to create stealthy and effective backdoor triggers for graph classification models.
- We introduce a dynamic poisoned sample selection strategy based on the occurrence of forgetting events, enabling the selection of poisoned samples that contribute more effectively to the backdoor attack.
- We conducted attack evaluations on GCN and GIN models across four benchmark datasets. The results demonstrate that our approach achieves better attack performance while maintaining a low clean accuracy drop (CAD).

2 Related Work

Backdoor attacks in graph classification remain an emerging area of research, with significant challenges yet to be addressed. Early research by [21] pioneered the idea of injecting predefined fixed subgraph patterns into training samples, prompting the GNN model to associate these subgraphs with attacker-designated target labels. Based on this, [18] further explored how the injection position of trigger subgraphs affects the attack performance, revealing that strategic placement at critical positions significantly influences attack success. [15] introduced an adaptive trigger generation approach using neural networks, producing triggers with fixed node counts but varied topologies, thereby enhancing the attack's flexibility and reducing detectability. [19] proposed perturbation-based triggers, leveraging meta-gradients computed on adjacency matrices to guide structural modifications. More recently, [22] investigated how different subgraph motifs influence the effectiveness of backdoor attacks from a motif-based perspective. Despite these advancements, existing methods have largely ignored the critical role of sample-level variations in backdoor attack effectiveness, leaving room for more efficient and stealthy approaches. We summarize and compare mainstream backdoor attack methods on graph classification tasks in Table 1.

3 Preliminaries and Problem Definition

3.1 Graph Classification and GNNs

A graph classification task operates on input graphs $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the node set with feature vectors $\{\mathbf{x}_i\}_{v_i \in \mathcal{V}}$ and $\mathcal{E}$ represents the set of edges capturing pairwise relationships between nodes. The goal is to learn a function $f : G \to y_G$ that maps structural and attribute information to a graph-level label y_G.

Modern approaches leverage GNNs, which iteratively refine node embeddings through neighborhood aggregation. At layer l, the embedding $\mathbf{h}_i^{(l)}$ of node v_i is updated via:

$$\mathbf{h}_i^{(l)} = \text{COMBINE}^{(l)}\left(\mathbf{h}_i^{(l-1)}, \text{AGGREGATE}^{(l)}\left(\{\mathbf{h}_j^{(l-1)} \mid j \in \mathcal{N}(i)\}\right)\right),$$

where $\mathcal{N}(i)$ is the neighbor set of v_i, $\text{AGGREGATE}^{(l)}(\cdot)$ integrates neighboring embeddings, and $\text{COMBINE}^{(l)}(\cdot)$ fuses aggregated features with the node's current state.

To obtain graph-level predictions, a readout function aggregates the final-layer node embeddings $\{\mathbf{h}_i^{(L)}\}_{v_i \in \mathcal{V}}$ into a global graph representation:

$$\mathbf{z}_G = \text{READOUT}\left(\{\mathbf{h}_i^{(L)}\}\right),$$

where $\text{READOUT}(\cdot)$ is typically a pooling or summarization function that aggregates node-level embeddings from the final layer L into a single vector representation for the entire graph. The resulting graph embedding $\mathbf{z}_G$ is then fed into a classifier (e.g., multi-layer perceptron) to predict y_G.

This architecture enables GNNs to capture both *local structural patterns* (via neighborhood aggregation) and *global graph characteristics* (via readout operations), forming the foundation for modern graph classification systems.

3.2 Problem Formulation

Attacker's Knowledge and Capability. We analyze two attack scenarios, considering both white-box and black-box settings. In the white-box attack (as demonstrated in our experiments on the GCN model), the attacker employs a surrogate model with the same architecture as the victim model, enabling a more precise simulation of the target's behavior and thereby enhancing the attack's effectiveness. In contrast, the black-box attack (evaluated using our method on the GIN model) involves a surrogate model with a different architecture from the victim model, preventing the attacker from directly accessing the target's parameters or structural details. Instead, the attack relies on transferring adversarial patterns learned from the surrogate model. In both cases, the attacker manipulates a small portion of the training data by injecting structural or feature-based trigger patterns and modifying the corresponding labels, leading the victim model to misclassify inputs during inference.

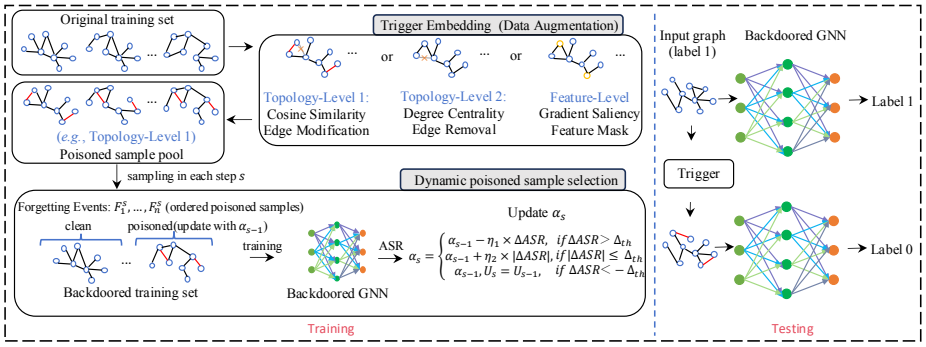


Fig. 1: The framework of our backdoor attack on graph classification task.

Adversary's Objective. Formally, given a GNN classifier F, a targeted attack class y_t, and a trigger graph denoted by G_{g_t}, the adversary aims to achieve two main objectives:

$$\begin{cases} F(G_{g_t}) = y_t, & \text{for trigger-embedded graphs } G_{g_t}, \\ F(G) = F_o(G), & \text{for clean graphs } G. \end{cases} \tag{1}$$

The first objective ensures the attack effectiveness: whenever the trigger g_t is embedded, the trojaned model F consistently classifies the input graph into the attacker-specified target class y_t. The second objective ensures attack evasiveness: the trojaned model F behaves identically to a clean model F_o on clean graphs, making the backdoor difficult to detect.

4 Method

We propose a novel backdoor attack framework for graph classification that comprises two designs: trigger construction via graph augmentation and dynamic poisoned sample selection based on forgetting events. As illustrated in Figure 1, the attack process begins by applying a data augmentation-based strategy to generate poisoned samples, where each clean graph is perturbed using one of three predefined augmentation methods. Specifically, these methods include two topology-level strategies and one feature-level strategy. To preserve stealthiness, only one augmentation strategy is applied to each poisoned sample at a time.

Following augmentation, we adopt a dynamic poisoned sample selection strategy, which adaptively filters out low-impact poisoned samples and adjusts the selection process based on real-time ASR feedback. This enables the identification of high-value samples that maximize the effectiveness of the backdoor attack. During testing, we apply our designed trigger to clean samples, causing them to be classified as the target class by the backdoored GNN.

In the following sections, we first describe the three augmentation strategies used to construct backdoor triggers, and then detail our dynamic poisoned sample selection strategy.

4.1 Trigger Construction via Graph Augmentation

Topology-Level Augmentation. At the graph structure level, we employ two edge modification strategies.

Strategy 1: Edge Modification Based on Cosine Similarity. We utilize the cosine similarity between node feature vectors to guide edge addition and removal, ensuring that the modified edges blend naturally into the local structure. For any two unconnected nodes v_i and v_j, the cosine similarity between their feature vectors is computed as follows:

$$\mathrm{Sim}(v_i, v_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}. \tag{2}$$

Based on these similarity values, we rank all possible node pairs. The top-k edges with the lowest similarity scores are removed if they exist, while the top-k edges with the highest similarity scores are added if they do not exist in the original graph.

Strategy 2: Edge Removal Based on Degree Centrality. To preserve the core structural and semantic information of the graph, we identify and remove edges with lower importance, as proposed in [23]. Specifically, we assign each existing edge an associated removal probability $p_{v_i v_j}$, reflecting its importance. We use the degree centrality of edges as the measure of their importance. According to the definition proposed in [23], the edge centrality based on node degree centrality is defined as the average of its two endpoints:

$$w_{v_i,v_j} = \frac{c(v_i) + c(v_j)}{2}, \tag{3}$$

where $c(v_i)$ and $c(v_j)$ denote the degree centrality values of the endpoints v_i and v_j, calculated as follows:

$$c(v_j) = \frac{\deg(v_j)}{|\mathcal{N}| - 1}, \tag{4}$$

with $\mathcal{N}$ representing the total number of nodes in the graph.

We apply a logarithmic transformation to the edge centrality to mitigate the dominant effect of high-degree nodes:

$$s_{v_i,v_j} = \log(w_{v_i,v_j}). \tag{5}$$

Finally, the edge removal probability is normalized to:

$$p_{v_i v_j} = \left(\frac{s_{\max} - s_{v_i,v_j}}{s_{\max} - s_{\min}} \right) \cdot p_e, \tag{6}$$

where p_e controls the overall probability of edge removal, $s_{\max}$ and $s_{\min}$ denote the maximum and minimum values of s_{v_i,v_j} respectively.

Feature-Level Augmentation. At the feature level, we employ the feature masking strategy.

Strategy 3:Feature Masking. For feature-level augmentation, we adopt a selective feature masking strategy based on degree centrality and gradient saliency masks, allowing us to protect critical features while perturbing less important ones.

First, we leverage degree centrality to distinguish between critical and non-critical nodes. High-centrality nodes retain all their features to preserve global information, whereas non-critical nodes undergo selective feature masking. We select the bottom $\rho\%$ of nodes with the lowest degree centrality for feature masking, ensuring that structurally less critical nodes undergo perturbation while preserving key graph information.The importance of each feature is determined using a gradient saliency mask, computed by backpropagating the gradients with respect to the node feature vector $\mathbf{x}_v$:

$$\mathbf{g}_v = \left| \frac{\partial \mathcal{L}}{\partial \mathbf{x}_v} \right|, \tag{7}$$

where $\mathcal{L}$ represents the model's loss function, and $\mathbf{g}_v$ measures the contribution of each feature dimension to the classification task. We then rank feature dimensions based on their importance scores and retain the top $(1 - q\%)$ of features while masking the remaining $q\%$ ones:

$$\tilde{\mathbf{x}}_v = \mathbf{x}_v \odot \mathbf{m}, \tag{8}$$

where $\mathbf{m} \in \{0, 1\}^F$ is the feature mask vector, defined as:

$$m_i = \begin{cases} 1, & i \in \text{Top-}(1 - q\%)(\mathbf{g}_v), \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

This method ensures that key discriminative features remain intact while introducing subtle noise by masking less significant features, making the attack more difficult to detect.

4.2 Dynamic Poisoned Sample Selection by Forgetting Events

The quality of poisoned samples directly influences the success rate and stealthiness of backdoor attacks. Traditional random sample selection methods often introduce redundant or low-impact samples, thereby reducing the ASR and increasing the risk of detection. Our objective is to identify a subset of samples that contribute significantly to the backdoor attack. Prior works [6,13] have indicated that forgetting samples reveal intrinsic data properties and play a crucial role in shaping the classifier's decision boundary, whereas samples that are rarely forgotten have a limited impact on the final model performance. [16] explored the influence of forgetting samples in image-based backdoor attacks and proposed the Filtering-and-Updating Strategy (FUS) to refine sample selection. Building upon this foundation, we investigate the effect of forgetting backdoor samples

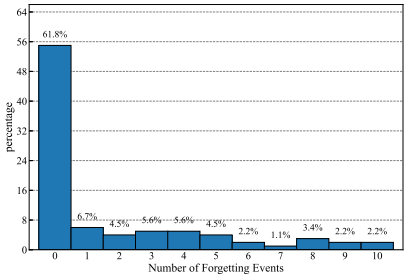


Fig. 2: Number of forgetting events on the PROTEINS dataset.

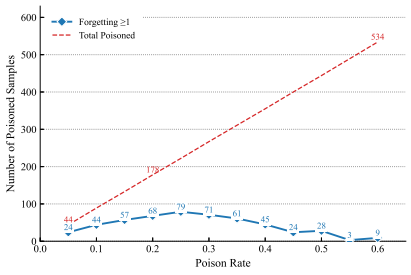


Fig. 3: Number of forgetting samples under different poisoning rates.

in graph-based data and enhance the FUS strategy by introducing a dynamic adjustment mechanism that adapts in real-time to sample performance, thereby efficiently selecting the most impactful poisoned samples for the attack.

Definition of Forgetting Events. A forgetting event occurs when a model correctly classifies a sample at the time step r but subsequently misclassifies it in the time step $r+1$. Formally, the forgetting count F_n of poisoned sample n is defined as:

$$F_n = \sum_{r=1}^{R-1} \mathbb{I}(\hat{y}_n^r = y_t \text{ and } \hat{y}_n^{r+1} \neq y_t), \tag{10}$$

where y_t represents the target label, $\hat{y}_n^r$ denotes the model's predicted label for sample n at the r-th training epoch, R denotes the total number of training epochs, and $\mathbb{I}(\cdot)$ is the indicator function.

Figure 2 illustrates an experiment conducted on the PROTEINS dataset. In this experiment, 61.8% of the data did not experience any forgetting events, while 38.2% of the data underwent at least one forgetting event. We refer to samples that have experienced at least one forgetting event as forgetting samples. [13] suggested that samples with frequent forgetting events contribute more significantly to the learning of backdoor patterns.

We further measure the number of forgetting samples under different poisoning rates, with the results shown in Figure 3. It can be observed that as the poisoning rate increases, the number of forgetting samples initially increases and then decreases. Given a fixed poisoning rate, we aim to identify the most valuable forgetting samples for poisoning. We can achieve this goal by greedily retaining a large sample set of F_n, but it should be noted that only a small portion of the poisoned samples are recorded each time, and the forgetting events of the remaining clean training samples are not recorded. Therefore, this greedy retention method is local. According to the trend of change shown in Figure 3, it is also infeasible to achieve this by expanding the poisoning set.

[13] employed the FUS strategy for image data sample selection. The FUS operates as follows: first, all training samples are poisoned to construct the poi-

soned sample pool $\mathcal{D}_{poisoned}$. The attacker randomly selects $U' \leftarrow p \cdot \mathcal{D}_{poisoned}$ samples from this poisoned sample pool based on a predefined poisoning rate p. The training process then iterates for S rounds, where in each round, a fixed proportion $(\alpha \cdot |U'|)$ of the lowest-impact samples, ranked by F_n, are removed. Subsequently, an equal number $(\alpha \cdot |U'|)$ of new samples are randomly selected from the $\mathcal{D}_{poisoned} \setminus U'$. This iterative process continues until completion.

However, the FUS strategy has limitations: the filtering ratio α is fixed, and sample selection relies solely on direct removal without dynamic adjustment based on real-time sample performance. To address these issues, we propose a Dynamically Adjusted Filtering-and-Updating Strategy (DAFUS). We observe that the same poisoned sample may exhibit varying F_n when trained with different poisoned sample pools. Moreover, because we employ an early stopping mechanism during training, the number of training epochs can further influence the F_n value of a given poisoned sample. Therefore, instead of adjusting based on the average number of forgetting samples, we dynamically adapt the selection based on the attack success rate. We define the change in ASR as:

$$\Delta\text{ASR} = \text{ASR}_s - \text{ASR}_{s-1}. \tag{11}$$

The update ratio α for poisoned samples is dynamically adjusted as follows:

$$\alpha_s = \begin{cases} \alpha_{s-1} - \eta_1 \times (\Delta\text{ASR}), & \text{if } \Delta\text{ASR} > \Delta_{\text{th}}, \\ \alpha_{s-1} + \eta_2 \times (|\Delta\text{ASR}|), & \text{if } |\Delta\text{ASR}| \leq \Delta_{\text{th}}, \\ \alpha_{s-1}, \quad U_s = U_{s-1}, & \text{if } \Delta\text{ASR} < -\Delta_{\text{th}}. \end{cases} \tag{12}$$

where Δ_{th} represents a predefined threshold that distinguishes significant changes in ASR, η_1 and η_2 are adjustment rates that determine the magnitude of adjustments, and U_s denotes the backdoor training set at iteration s.

- When the ASR increases significantly ($\Delta\text{ASR} > \Delta_{\text{th}}$), the update ratio α is proportionally reduced based on the increase in ASR. This ensures that high-contribution samples are retained, preventing excessive replacement.
- When the change in ASR is insignificant ($|\Delta\text{ASR}| \leq \Delta_{\text{th}}$), the update ratio α increases proportionally according to the magnitude of the variation in ASR. This enhances exploration and mitigates the risk of being trapped in a local optimum.
- When the ASR decreases significantly ($\Delta\text{ASR} < -\Delta_{\text{th}}$), the poisoned sample set is immediately reverted to the previous iteration ($U_s = U_{s-1}$). This rollback mechanism helps restore attack effectiveness, which may have been compromised due to the removal of high-contribution samples.

We maintain an index of the poisoned samples for each iteration and ultimately select the sample set corresponding to the highest ASR as the final poisoning set. This strategy effectively balances exploration and exploitation by adaptively adjusting the update ratio of poisoned samples, leading to improved attack performance. We summarized the overall process of our attack framework in Algorithm 1.

Algorithm 1 Our backdoor attack framework

Input: Clean training set $\mathcal{D}_{\text{clean}}$, Poisoning rate p, Initial filter ratio α_0, Threshold Δ_{th}, Adjustment rates η_1, η_2, Max iterations S, Target class y_t
Output: Constructed poisoned training set $\mathcal{U}$
 # Data Augmentation-based Triggers
Generate poisoned sample pool $\mathcal{D}_{\text{poisoned}} \leftarrow \{(\text{augmentation}(G_i), y_t) \mid G_i \in \mathcal{D}_{\text{clean}}\}$ from topology-level or feature-level augmentation strategy
 # Dynamic poisoned sample selection
Initialize backdoor training set U from $\mathcal{D}_{\text{clean}}$ and $U' \leftarrow p \cdot \mathcal{D}_{\text{poisoned}}$
for $s = 1$ to S **do**
  Train model on U, record forgetting count $\{F_1, ..., F_n\}$ via Eq. 10
  Compute ASR_s and then calculate ΔASR via Eq. 11
  update filter ratio α_s via Eq. 12
  Sort U' by descending F_n
  Filter $\alpha_s \cdot |U'|$ samples with lowest F_n
  Randomly replenish same number from $\mathcal{D}_{\text{poisoned}} \setminus U'$
  update backdoor training set U
end for

5 Experiments

5.1 Experimental Setup

Datasets. We conduct experiments on four widely used graph classification datasets: PROTEINS, NCI1, Mutagenicity and ENZYMES. We split each dataset into 75% for training, 5% for validation, and 20% for testing. We summarize the key statistics of the datasets used in our experiments in Table 2.
Evaluation Metrics. We assess the effectiveness and evasiveness of our attack using the following two key metrics:

- **ASR**: The percentage of instances in the test set, originally belonging to non-target classes, that are successfully misclassified into the target class after embedding the backdoor trigger.
- **CAD**: The decrease in accuracy on clean test samples compared to a model trained without poisoning.

Hyperparameter Settings. We utilize two GNNs (GCN and GIN), each configured with a two-layers structure and ReLU activation function. The models are trained using the Adam optimizer with a learning rate of 0.01 and a weight decay of 5e-4. The maximum number of training epochs is set to 100, with an early stopping mechanism (patience=10) to prevent overfitting. During training, we use a batch size of 32 and apply a dropout rate of 0.1 for regularization. Table 3 shows the clean model accuracy of GCN and GIN on different datasets.

For the backdoor attack, we set the poisoning rate to 10%. In Data Augmentation Strategy 1, we set k=2. In Strategy 2, the edge removal probability p_e is set to 0.8. In Strategy 3, we select the bottom 30% of nodes based on degree

Table 2: Summary of benchmark datasets.

Dataset	Graphs	Avg. Nodes	Avg. Edges	Classes
NCI1	4,110	29.8	32.3	2
PROTEINS	1,113	39.1	72.8	2
Mutagenicity	4,337	30.3	30.8	2
ENZYMES	600	32.6	62.1	6

Table 3: Clean model accuracy of GCN and GIN

Dataset	GCN	GIN
PROTEINS	0.7130	0.7264
NCI1	0.7251	0.7615
Mutagenicity	0.7707	0.8099
ENZYMES	0.3583	0.3833

centrality, and 15% of their features are masked. For the poisoned sample selection mechanism, we set the initial α to 0.5, the number of iterations S to 40, the ASR rollback threshold Δ_{th} to 0.05, and the adjustment rates η_1 and η_2 to 1.

For the baseline methods we compared, we adopt the parameter configuration specified in their original implementation.

Baseline Methods. Subgraph [21]: This method inserts a randomly generated fixed subgraph into a subset of training samples and assigns them to a target label. The trigger subgraph remains consistent across all poisoned samples. **Motif** [22]: This method utilizes frequent subgraph motifs as triggers, making them more naturally embedded in real-world datasets. **TRAP** [19]: TRAP flips edges in poisoned samples based on meta-gradients to create imperceptible perturbations that influence model predictions. **GTA** [15]: This method generates adversarial subgraphs using neural networks and injects them into poisoned samples, enhancing the attack's flexibility while maintaining stealthiness.

Our approach is named **Ours-Cos** when utilizing the first data augmentation strategy, **Ours-Deg** for the second, and **Ours-Mas** for the third.

5.2 Overall performance of backdoor attacks

We use GCN as the backdoor model for trigger injection and sample selection. To evaluate our method, we measure the ASR and CAD on GCN models with the same architecture, representing a white-box attack scenario. Additionally, we assess the transferability of our attack by testing it on GIN models, which correspond to black-box attack settings.

Table 4 presents the experimental results. Among our three proposed strategies, Ours-Cos and Ours-Deg consistently achieve the highest ASR, while maintaining competitive or even lower CAD compared to baseline methods. This indicates that our method achieves better attack performance with comparable evasiveness to the baseline.

In comparison, Ours Cos has a slightly higher CAD, while Ours Deg achieves a better balance between ASR and CAD. Ours-Mas, which perturbs features rather than structure, shows the lowest ASR among our three strategies but still surpasses some baselines like Subgraph Backdoor and TRAP. This result suggests that structural perturbations are more effective than feature perturbations

Table 4: Comparison of Attack Performance on GCN and GIN.

Method	PROTEINS		NCI1		Mutagenicity		ENZYMES	
	ASR	CAD	ASR	CAD	ASR	CAD	ASR	CAD
Results on GCN								
Subgraph Backdoor	0.7263	0.0035	0.7288	0.0011	0.7698	0.0046	0.5254	**0.0083**
Motif	0.8842	0.0538	0.9733	0.0213	0.8544	0.0196	0.7288	0.0167
TRAP	0.8421	0.0179	0.8305	0.0284	0.8148	0.0276	0.6779	0.0250
GTA	0.8211	-0.0069	0.9612	0.0314	0.8968	0.0104	0.6949	0.0167
Ours-Cos	0.9474	0.0483	0.9758	0.0257	**0.9735**	0.0419	**0.8305**	0.0416
Ours-Deg	**0.9578**	0.0046	**0.9831**	0.0049	0.9656	**-0.0081**	0.7966	**0.0083**
Ours-Mas	0.9263	0.0404	0.9346	0.0173	0.9497	0.0150	0.7457	0.0167
Results on GIN								
Subgraph Backdoor	0.7215	-0.0011	0.7275	0.0018	0.7751	0.0042	0.5084	0.0083
Motif	0.8813	0.0571	0.9735	0.0199	0.8518	0.0191	0.7118	0.0167
TRAP	0.8401	0.0134	0.8306	0.0271	0.8201	0.0275	0.6613	0.0333
GTA	0.8184	0.0028	0.9686	0.0804	0.8941	0.0121	0.8779	0.0250
Ours-Cos	0.9443	0.0458	0.9782	0.0346	**0.9682**	0.0562	**0.8805**	0.0416
Ours-Deg	**0.9539**	0.0092	**0.9806**	0.0064	**0.9682**	0.0099	0.7796	**0.0083**
Ours-Mas	0.9225	0.0377	0.9370	0.0157	0.9523	0.0162	0.7288	0.0167

in attacking graph models, likely due to the greater impact of structural changes on model behavior.

Among the baseline methods, Motif and GTA perform well in ASR, while TRAP and Subgraph Backdoor exhibit moderate performance, indicating that the model does not effectively learn strong trigger patterns from these methods. However, due to its relatively moderate attack performance, its CAD is also correspondingly lower.

An interesting observation is that some methods, such as GTA, Subgraph Backdoor, and Ours-Deg, achieve negative CAD values on certain datasets. This indicates that, in some cases, the poisoned model slightly improves rather than degrades the clean accuracy. This phenomenon may be due to the way poisoned samples shift the decision boundary, inadvertently leading to a regularization effect that enhances model generalization.

5.3 Impact of poisoning rate

To analyze the impact of varying poisoning rates on the ASR, we evaluate three attack strategies of the GCN model on the NCI1 dataset under seven different poisoning rates: 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, and 0.14. The experimental results are presented in Figure 4.

As expected, increasing the poisoning rate consistently improves the ASR for all attack strategies. This is because a higher poisoning rate introduces more backdoored samples into the training process, allowing the model to learn the trigger pattern more effectively.

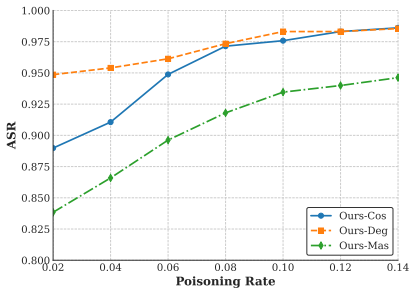


Fig. 4: Sensitivity of ASR to different poisoning rates.

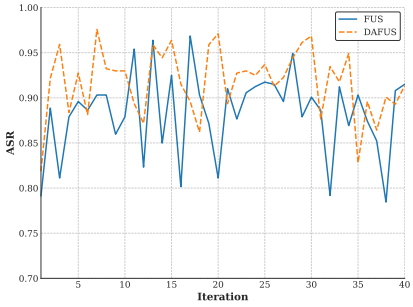


Fig. 5: Sensitivity of ASR to different sample selection strategies.

Among the three strategies, Ours-Deg demonstrates the best overall performance. Even at a low poisoning rate of 0.02, it maintains a high attack success rate, and as the poisoning rate increases, its ASR stabilizes, indicating its robustness in embedding backdoor triggers efficiently.

Ours-Cos performs slightly worse than Ours-Deg at lower poisoning rates but exhibits a rapid improvement as the poisoning rate increases. At 0.14, its ASR surpasses Ours-Deg, suggesting that cosine similarity-based modifications provide a more scalable and consistent impact across different contamination levels. Ours-Mas consistently shows the lowest ASR across all poisoning rates but follows an upward trend, indicating that feature-masking-based backdoors are effective, albeit requiring a higher poisoning rate to achieve performance comparable to structure-based triggers.

Notably, even at lower poisoning rates, our three attack strategies outperform some baseline methods at a 0.10 poisoning rate in Table 4, particularly TRAP and Subgraph Backdoor. This is attributed to our sample selection mechanism, which ensures that even with a low poisoning rate, only the most impactful samples are chosen for poisoning, thereby maintaining high ASR. This selective approach enables our method to achieve effective backdoor attacks with fewer poisoned samples, improving stealthiness while preserving attack performance.

5.4 Ablation experiment

Impact of Poisoned Sample Selection Strategy. To evaluate the impact of different sample selection strategies on attack performance, we conducted experiments using three approaches: DAFUS, FUS, and Random Selection. Cosine similarity-based edge modifications from Strategy 1 were employed as the backdoor trigger. We measured their ASR performance on both GCN and GIN models, with the results summarized in Table 5. DAFUS consistently outperforms both FUS and RANDOM. Compared to RANDOM, DAFUS improves ASR by 3.4%–9.7%, while FUS achieves an ASR improvement of 1.4%–5.3%. This demonstrates that forgetting-event-based poisoned sample selection effec-

Table 5: Comparison of ASR under different sample selection strategies

Sample Selection Strategy	PROTEINS		NCI1		Mutagenicity		ENZYMES	
	GCN	GIN	GCN	GIN	GCN	GIN	GCN	GIN
DAFUS	**0.9474**	**0.9443**	**0.9758**	**0.9782**	**0.9735**	**0.9682**	**0.8305**	**0.8805**
FUS	0.9023	0.8910	0.9104	0.9507	0.9456	0.9332	0.8105	0.8223
RANDOM	0.8761	0.8654	0.8886	0.9065	0.8924	0.8801	0.7967	0.7839

Table 6: Comparison of ASR under different Trigger

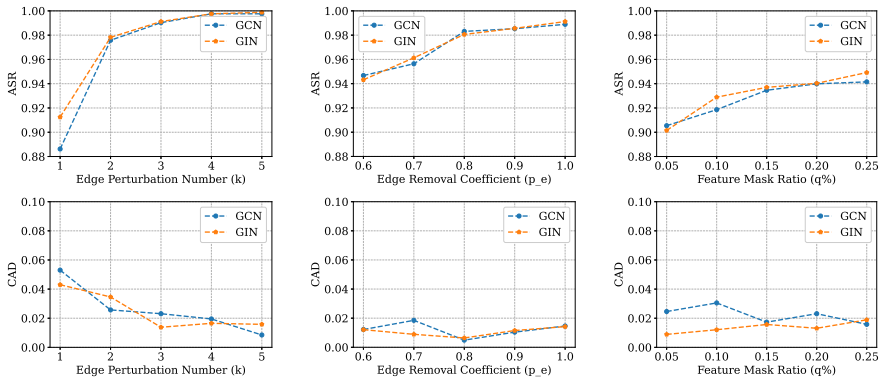
Augmentation Strategy	PROTEINS		NCI1		Mutagenicity		ENZYMES	
	GCN	GIN	GCN	GIN	GCN	GIN	GCN	GIN
Ours-Cos	0.9474	0.9443	0.9758	0.9782	**0.9735**	**0.9682**	**0.8305**	**0.8805**
Ours-Deg	**0.9578**	**0.9539**	**0.9831**	**0.9806**	0.9656	**0.9682**	0.7966	0.7796
Ours-Mas	0.9263	0.9225	0.9346	0.9370	0.9497	0.9523	0.7457	0.7288
Random-Edge	0.7676	0.7701	0.7789	0.7251	0.7265	0.6853	0.5103	0.5397
Random-Mask	0.6135	0.5846	0.5122	0.5346	0.6154	0.6452	0.4922	0.4135

tively identifies more impactful poisoned samples for backdoor attacks. Furthermore, DAFUS outperforms FUS by an additional 2.0%–6.5%, highlighting that our dynamic adjustment mechanism enhances the efficiency and effectiveness of selecting critical samples.

Figure. 5 illustrates the differences between FUS and DAFUS during iterations. As observed, FUS exhibits irregular fluctuations, with ASR varying unpredictably. In contrast, DAFUS employs a dynamic update rate and effectively rolls back samples when the attack success rate declines. This adaptive mechanism enables DAFUS to efficiently select the most relevant poisoned sample set based on real-time sample performance.

Impact of Data Augmentation Methods. To evaluate the effectiveness of our proposed data augmentation-based backdoor trigger, we compare it with two baseline augmentation strategies: Random-Edge (randomly modifying edges) and Random Feature Masking (randomly selecting nodes and applying random feature masks). The hyperparameter settings for these two baselines match those of Ours-Cos and Ours-Mas, respectively. All methods adopt the DAFUS strategy.

From the results in Table 6, we observe that our proposed backdoor attacks, based on three different data augmentation strategies, consistently outperform all baseline methods. This demonstrates that our data augmentation-based triggers effectively embed the trigger pattern into the model.



(a) Our-Cos (b) Our-Deg (c) Our-Mas

Fig. 6: Comparison of Different trigger parameters.

5.5 Impact of trigger parameters.

In this section, we analyze the impact of trigger parameters (i.e., the amplitude of perturbation on a single sample) on experimental results. We conducted this experiment on the NCI1 dataset.

In Figure 6, it can be observed that the ASR of the three methods exhibits a relatively consistent trend. Specifically, when the perturbation magnitude on the graph structure is small, the ASR remains relatively low. As the perturbation increases, the ASR rises rapidly in the initial stage and then gradually stabilizes. In contrast, the CAD does not follow a uniform pattern between the three methods. In most experimental settings, the CAD remains below 0.05, suggesting that these methods achieve high attack success rates while exerting minimal impact on the classification performance of clean samples.

6 Conclusion

This paper proposes a novel backdoor attack framework for graph classification. Our method introduces three alternative augmentation strategies and designs a sample selection mechanism to enhance attack efficiency and stealthiness.

Comprehensive experiments on four benchmark datasets demonstrate that our method outperforms existing backdoor attacks. Our approach achieves a higher ASR while maintaining a relatively low CAD. Among our three trigger strategies, Ours-Deg consistently balances ASR and CAD.

Overall, our study highlights the vulnerability of graph classification models to data augmentation-based backdoor attacks. Future work will focus on refining augmentation strategies and exploring more effective sample selection techniques to further improve attack efficiency and stealthiness.

Acknowledgments. This research was funded by the National Key R&D Program of China (Grant No.2024YFE0209000), the NSFC (Grant No.U23B2019).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bi, B., Zhang, Z., Qiao, P., Yuan, Y., Wang, G.: Fedsig: A federated graph augmentation for class-imbalanced node classification. In: International Conference on Database Systems for Advanced Applications. pp. 474–490. Springer (2024)
2. Corso, G., Stark, H., Jegelka, S., Jaakkola, T., Barzilay, R.: Graph neural networks. Nature Reviews Methods Primers **4**(1), 17 (2024)
3. Dai, E., Lin, M., Zhang, X., Wang, S.: Unnoticeable backdoor attacks on graph neural networks. In: Proceedings of the ACM Web Conference 2023. pp. 2263–2273 (2023)
4. Ding, K., Xu, Z., Tong, H., Liu, H.: Data augmentation for deep graph learning: A survey. ACM SIGKDD Explorations Newsletter **24**(2), 61–77 (2022)
5. Ianni, M., Masciari, E., Sperlí, G.: A survey of big data dimensions vs social networks analysis. Journal of Intelligent Information Systems **57**, 73–100 (2021)
6. Katharopoulos, A., Fleuret, F.: Not all samples are created equal: Deep learning with importance sampling. In: International conference on machine learning. pp. 2525–2534. PMLR (2018)
7. Li, Y., Jiang, Y., Li, Z., Xia, S.T.: Backdoor learning: A survey. IEEE transactions on neural networks and learning systems **35**(1), 5–22 (2022)
8. Liu, Y., Ma, X., Bailey, J., Lu, F.: Reflection backdoor: A natural backdoor attack on deep neural networks. In: Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, part X 16. pp. 182–199. Springer (2020)
9. Qiao, P., Zhang, Z., Li, Z., Zhang, Y., Bian, K., Li, Y., Wang, G.: Tag: Joint triple-hierarchical attention and gcn for review-based social recommender system. IEEE Transactions on Knowledge and Data Engineering **35**(10), 9904–9919 (2022)
10. Richards, A.L., Eckhardt, M., Krogan, N.J.: Mass spectrometry-based protein–protein interaction networks for the study of human diseases. Molecular systems biology **17**(1), e8792 (2021)
11. Rodríguez-Pérez, R., Bajorath, J.: Evolution of support vector machine and regression modeling in chemoinformatics and drug discovery. Journal of Computer-Aided Molecular Design **36**(5), 355–362 (2022)
12. Sui, Y., Wang, S., Sun, J., Liu, Z., Cui, Q., Li, L., Zhou, J., Wang, X., He, X.: A simple data augmentation for graph classification: A perspective of equivariance and invariance. ACM Transactions on Knowledge Discovery from Data **19**(2), 1–24 (2025)
13. Toneva, M., Sordoni, A., Combes, R.T.d., Trischler, A., Bengio, Y., Gordon, G.J.: An empirical study of example forgetting during deep neural network learning. arXiv preprint arXiv:1812.05159 (2018)
14. Wu, L., Cui, P., Pei, J., Zhao, L., Guo, X.: Graph neural networks: foundation, frontiers and applications. In: Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining. pp. 4840–4841 (2022)

15. Xi, Z., Pang, R., Ji, S., Wang, T.: Graph backdoor. In: 30th USENIX security symposium (USENIX Security 21). pp. 1523–1540 (2021)
16. Xia, P., Li, Z., Zhang, W., Li, B.: Data-efficient backdoor attacks. arXiv preprint arXiv:2204.12281 (2022)
17. Xia, X.: Bioinformatics and drug discovery. Current topics in medicinal chemistry **17**(15), 1709–1726 (2017)
18. Xu, J., Xue, M., Picek, S.: Explainability-based backdoor attacks against graph neural networks. In: Proceedings of the 3rd ACM workshop on wireless security and machine learning. pp. 31–36 (2021)
19. Yang, S., Doan, B.G., Montague, P., De Vel, O., Abraham, T., Camtepe, S., Ranasinghe, D.C., Kanhere, S.S.: Transferable graph backdoor attack. In: Proceedings of the 25th international symposium on research in attacks, intrusions and defenses. pp. 321–332 (2022)
20. Zeng, Y., Park, W., Mao, Z.M., Jia, R.: Rethinking the backdoor attacks' triggers: A frequency perspective. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 16473–16481 (2021)
21. Zhang, Z., Jia, J., Wang, B., Gong, N.Z.: Backdoor attacks to graph neural networks. In: Proceedings of the 26th ACM symposium on access control models and technologies. pp. 15–26 (2021)
22. Zheng, H., Xiong, H., Chen, J., Ma, H., Huang, G.: Motif-backdoor: Rethinking the backdoor attack on graph neural networks via motifs. IEEE Transactions on Computational Social Systems **11**(2), 2479–2493 (2023)
23. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Graph contrastive learning with adaptive augmentation. In: Proceedings of the web conference 2021. pp. 2069–2080 (2021)