Graph Defense Diffusion Model

Xin He
hex23@mails.jlu.edu.cn
Jilin University
China

Wenqi Fan
wenqifan03@gmail.com
The Hong Kong Polytechnic University
Hong Kong

Yili Wang
wangyili@jlu.edu.cn
Jilin University
China

Chengyi Liu
chengyi.liu@connect.polyu.hk
The Hong Kong Polytechnic University
Hong Kong

Rui Miao
miaorui24@mails.jlu.edu.cn
Jilin University
China

Xin Juan
junxin22@mails.jlu.edu.cn
Jilin University
China

Xin Wang
xinwang@jlu.edu.cn
Jilin University
China

Abstract

Graph Neural Networks (GNNs) are highly vulnerable to adversarial attacks, which can greatly degrade their performance. Existing graph purification methods attempt to address this issue by filtering attacked graphs. However, they struggle to defend effectively against multiple types of adversarial attacks (e.g., targeted attacks and non-targeted attacks) simultaneously due to limited flexibility. Additionally, these methods lack comprehensive modeling of graph data, relying heavily on heuristic prior knowledge. To overcome these challenges, we introduce the **G**raph **D**efense **D**iffusion **M**odel (GDDM), a flexible purification method that leverages the denoising and modeling capabilities of diffusion models. The iterative nature of diffusion models aligns well with the stepwise process of adversarial attacks, making them particularly suitable for defense. By iteratively adding and removing noises (edges), GDDM effectively purifies attacked graphs, restoring their original structures and features. Our GDDM consists of two key components: (1) Graph Structure-Driven Refiner, which preserves the basic fidelity of the graph during the denoising process, and ensures that the generated graph remains consistent with the original scope; and (2) Node Feature-Constrained Regularizer, which removes residual impurities from the denoised graph, further enhancing the purification effect. By designing tailored denoising strategies to handle different types of adversarial attacks, we improve the GDDM's adaptability to various attack scenarios. Furthermore, GDDM demonstrates strong scalability, leveraging its structural properties to seamlessly transfer across similar datasets without retraining. Extensive experiments on three real-world datasets demonstrate that GDDM outperforms state-of-the-art methods in defending against various adversarial attacks, showcasing its robustness and effectiveness. The anonymous code is in https://github.com/hexin5515/GDDM.

CCS Concepts

• **Mathematics of computing** → **Graph algorithms**; • **Information systems** → **Data cleaning**.

Keywords

Graph Neural Networks, Adversarial Attack, Diffusion Model

ACM Reference Format:
Xin He, Wenqi Fan, Yili Wang, Chengyi Liu, Rui Miao, Xin Juan, and Xin Wang. 2026. Graph Defense Diffusion Model. In *Proceedings of the 32st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '26).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Graph data, characterized by its ability to capture complex relationships within real-world systems, has become increasingly prevalent across diverse domains, such as recommendation systems [5], biological networks [17, 23], and social sciences [3, 6]. Building on the potential of graph data, GNNs have emerged as a particularly effective approach to representation learning, leveraging message-passing mechanisms to aggregate both structural and feature information. This capability has enabled GNNs to achieve considerable success in downstream tasks such as node classification, link prediction, and graph classification [8]. However, it also makes them vulnerable to adversarial attacks that exploit their dependence on interconnected features and structures [16, 26].

Adversarial attacks introduce imperceptible modifications to graph structures or node features, leading to significant degradation in model performance. Such attacks can be categorized into targeted attacks [36], which focus on misclassifying specific nodes, and non-targeted attacks [24], which aim to disrupt the overall

KDD '26, Jeju, Korea

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

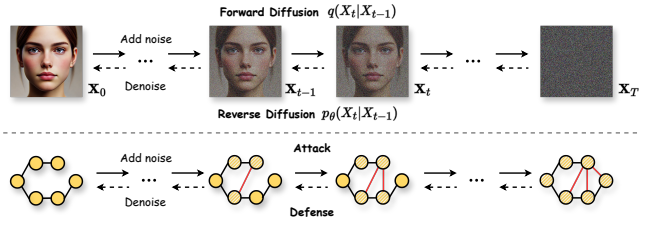


Figure 1: The connection between graph adversarial attack, defense and diffusion model.

graph integrity. The ability of these attacks to compromise model reliability poses serious challenges, especially in critical domains such as fraud detection [28], healthcare [32], and cyber security [20], where the consequences of incorrect predictions can be severe.

To mitigate the impact of adversarial attacks on GNNs, effective defense mechanisms are required to strengthen their robustness. Graph purification methods have shown promise by reconstructing attacked graphs without modifying the underlying GNN model. These methods are efficient and avoid the complexity of adversarial training routines [15]. However, current purification methods lack adaptability to diverse attack strategies [21], and depend heavily on heuristic-based prior knowledge rather than comprehensive modeling of the graph distribution [11, 36]. Diffusion models [7, 27] represent a promising strategy for addressing challenges associated with adversarial attacks on graphs, drawing on **their inherent consistency with the process of adversarial attack and defense**. As shown in Fig. 1, the forward diffusion process perturbs the data with random noise, stimulating the adversarial attacks to disrupt the graphs. Correspondingly, the reverse process reconstructs samples from noise, whose objective aligns well with the defense methods. This natural compatibility motivates us to **leverage the powerful denoising capabilities of diffusion models to defend against graph adversarial attacks uniformly.**

Although diffusion models hold potential for defending against graph adversarial attacks, their application remains unexplored due to three key challenges: #1 **Graph fidelity**: purifying high-fidelity graphs directly is difficult due to the complexity of graph topology and the inherent minor noise in the original graph [14, 30]. #2 **Localized denoising**: current diffusion models perform global denoising, making it hard to focus on specific nodes or edges affected by targeted attacks. #3 **Scalability**: current graph diffusion models [19, 35] suffer from high computational complexity, resulting in an unacceptable training cost on large graphs.

In this paper, we propose the **G**raph **D**efense **D**iffusion **M**odel (GDDM) to leverage the denoising power of graph diffusion models in defending against diverse adversarial attacks on graphs, aiming to address the three aforementioned challenges. Supervised by different levels of graph labels (e.g., adjacency matrix and node feature), GDDM implements targeted interventions on the current formation state to maximize its fidelity to the ground truth: During the denoising process, **Graph Structure-Driven Refiner (GSDR)** iteratively removes incorrect edges while preserving valid connections, ensuring basic fidelity. At the end of the denoising process, **Node Feature-Constrained Regularizer (NFCR)** uses node features to eliminate residual impurities, making precise corrections without altering the overall graph structure. The coupling of these two components provides reliable quality control throughout the entire denoising process, collaboratively ensuring graph fidelity (Challenge #1). We also design **Tailored Attack-specific Denoising Strategy (TADS)** to focus the model on denoising edges connected to target nodes, while retaining non-target edges as the foundation of the process, achieving localized denoising (Challenge #2). Furthermore, GDDM uses node degree information as input to model the graph generation process, and the degree distribution of nodes often exhibits similarity across different graph datasets. Therefore, GDDM can effortlessly transfer from smaller datasets to larger ones for denoising without retraining (Challenge #3).

The contributions of this work are summarized as follows:

- We propose the Graph Defense Diffusion Model (GDDM), a novel graph defense framework leveraging the natural alignment between the noise adding or denoising process of diffusion models and adversarial attack or defense to effectively defend against graph adversarial attacks.
- We introduce two key components (GSDR, NFCR) for high-fidelity denoised graphs and design tailored denoising strategies (TADS) to achieve localized denoising for targeted attacks. Besides, the similarity between graph datasets of the same type and the unique structure of the model contribute to GDDM's scalability across different datasets.
- Extensive experiments demonstrate that GDDM provides superior performance in defending against adversarial attacks, outperforming current state-of-the-art methods.

2 Related Work

2.1 Adversarial Attack on Graphs

Graph Neural Networks (GNNs) have been highly successful in graph data mining, but they are vulnerable to adversarial perturbations due to their dependency on the graph structure and message-passing mechanism [24]. One type of adversarial attack targets specific nodes in the graph, known as targeted attacks. Nettack [46] uses a greedy approach to perturb targeted nodes without altering node degree or feature co-occurrence. Another type of attack targets the entire graph, known as non-targeted attacks. Mettack [47] is a classic method using meta-learning for global graph attacks, but it is inefficient due to its focus on low-confidence nodes. GraD [24] improves the attack by focusing on medium-confidence nodes, enhancing efficiency. Basides, TDGIA [45] proposes a gradient-based node injection strategy that has shown stronger attack capability than traditional structure perturbation attacks. To support standardized evaluation of such attacks, GRB [43] introduces a unified benchmark covering multiple adversarial settings.

2.2 Graph Defense Methods

To counter adversarial attacks, various defense strategies have been proposed [13, 39, 42, 44]. One type of effective defense method is graph purification. Jaccard [36] defends against adversarial attacks by pruning edges based on the feature similarity between the two connected nodes. SVD [4] defends against adversarial attacks by removing the high-rank perturbations introduced by the attack methods. Guard [21] preemptively removes all potential noisy edges

by node degree information to protect target nodes. Another type of method is GNN-improved, which improves the robustness of GNNs by enhancing the GNN models themselves. ProGNN [15] and STABLE [22] mitigate the negative effects of adversarial attacks on GNNs by graph structure learning to filter out the noisy edges. DRAGON [40] proposes a scalable framework to address robustness degradation when scaling graph learning to large datasets. And GAME [41] focuses on model design and optimization strategies to improve the overall robustness of graph representations. Although existing defense methods achieve significant success, graph purification methods are unable to defend against multiple types of adversarial attacks.

2.3 Diffusion Models for Graph Data

Diffusion models [34, 38] have shown success in generating molecular and material graphs. GDSS [17] models the joint distribution of nodes and edges using stochastic differential equations to generate desired molecular graphs. DiGress [33] models a discrete diffusion process, performing graph denoising by changing types of edges at each recovery step. EDGE [2] models the change of each edge in the diffusion process as a Bernoulli distribution and introduces a guidance mechanism during recovery, significantly improving training efficiency and enabling the possibility of training diffusion models on large graphs. Although diffusion models excel at denoising, they have yet to be applied to adversarial graph defense. We observe a natural alignment between their processes and the dynamics of adversarial attacks and defenses, proposing a graph diffusion model with different denoising strategies guided by two fidelity enhancement components to defend against different types of graph adversarial attacks.

3 PRELIMINARY

3.1 Problem Statement

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ be an undirected graph, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of N nodes and $\mathcal{E}$ is the set of edges. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ represents the connectivity between nodes, where $\mathbf{A}_{ij} = 1$ if there is an edge between node v_i and node v_j, and $\mathbf{A}_{ij} = 0$ otherwise. The node feature matrix $\mathbf{X} \in \mathbb{R}^{N\times d}$ contains a d-dimensional feature vector for each node. Thus, a graph can also be compactly represented as $\mathcal{G} = (\mathbf{A}, \mathbf{X})$, where $\mathbf{A}$ encapsulates structural information and $\mathbf{X}$ represents node features. In the context of node classification tasks, adversarial attacks introduce perturbations primarily into the adjacency matrix $\mathbf{A}$ and sometimes into the feature matrix $\mathbf{X}$, which leads to a degradation in the performance of GNN classifiers. Let $\mathcal{G}' = (\mathbf{A}', \mathbf{X}')$ be the perturbed graph, where $\mathbf{A}'$ and $\mathbf{X}'$ denote the adversarially perturbed adjacency and feature matrices, respectively. In this work, we specifically consider the scenario where the feature matrix $\mathbf{X}$ remains unchanged, i.e., $\mathcal{G}' = (\mathbf{A}', \mathbf{X})$, and only the adjacency matrix $\mathbf{A}$ is compromised by adversarial edges. The goal is to employ a graph diffusion model to remove as many adversarial edges as possible from the perturbed graph $\mathcal{G}'$, effectively denoising the attacked graph while preserving its essential structure, thus improving the accuracy of downstream GNN classifiers in the node classification task.

3.2 Discrete Graph Diffusion and Recovery

The Discrete Graph Diffusion Model represents the noise injection process using a Binomial distribution [31], based on the assumption that each edge modification during noise injection is independent. In this setup, the adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ of the target graph $\mathcal{G}$ is a binary matrix with zeros on the diagonal. Let $\mathbf{A}^0$ denote the original adjacency matrix of the target graph. We then define the diffusion process as $q(\mathbf{A}^t|\mathbf{A}^{t-1})$ with t denoting the timestep. In the forward diffusion process, a Bernoulli distribution $\mathcal{B}(x;\mu)$ over the binary variable x with probability μ is used to represent the probability of each edge being resampled at the current timestep t:

$$\begin{aligned} q(\mathbf{A}^t|\mathbf{A}^{t-1}) &= \textstyle\prod_{i,j} \mathcal{B}(\mathbf{A}^t_{i,j}; (1-\beta_t)\mathbf{A}^{t-1}_{i,j} + \beta_t p) \\ &= \mathcal{B}(\mathbf{A}^t; (1-\beta_t)\mathbf{A}^{t-1} + \beta_t p), \end{aligned} \tag{1}$$

where β_t is the diffusion rate controlling the probability of resampling the element $\mathbf{A}^t_{i,j}$ from the Bernoulli distribution with probability p, rather than keeping the original $\mathbf{A}^t_{i,j}$. Therefore, this specific diffusion process allows direct sampling of $\mathbf{A}^t$ from $\mathbf{A}^0$:

$$q(\mathbf{A}^t|\mathbf{A}^0) = \mathcal{B}(\mathbf{A}^t; \bar{\alpha}_t\mathbf{A}^0 + (1-\bar{\alpha}_t)p), \tag{2}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. As t approaches infinity, $\bar{\alpha}_t$ converges to 0, meaning that the final adjacency matrix $\mathbf{A}^T$ becomes fully independent of the original $\mathbf{A}^0$. The value of p controls the probability of forming edges in $\mathbf{A}^T$, where a higher value of p implies a denser graph.

During the denoising process, the goal is to reverse the diffusion process by reconstructing $\mathbf{A}^{t-1}$ from $\mathbf{A}^t$, using a trained model to approximate the posterior distribution:

$$q(\mathbf{A}^{t-1}|\mathbf{A}^t, \mathbf{A}^0) = \frac{q(\mathbf{A}^t|\mathbf{A}^{t-1})q(\mathbf{A}^{t-1}|\mathbf{A}^0)}{q(\mathbf{A}^t|\mathbf{A}^0)}. \tag{3}$$

Since all terms are known, the model can directly compute Eq. 3 for recovering the original graph structure during training.

4 Graph Defense Diffusion Model

The architecture of the Graph Defense Diffusion Model (GDDM) is illustrated in Fig. 2, comprising two key components: the training phase and the inference phase. In the **training phase**, GDDM is trained on clean graph data leveraging a simplified discrete graph diffusion model. We reduce the model's complexity by employing sampling techniques for intermediate step calculations, enhancing training efficiency while maintaining the ability to reconstruct clean graph structures. During the **inference phase**, we introduce the graph structure-driven refiner and the node feature-constrained regularizer. They ensure high-fidelity denoised graph while enhancing GDDM's ability to defend against adversarial attacks. We incorporate tailored attack-specific denoising strategies to effectively address different types of adversarial attacks. Additionally, GDDM leverages the similarity in degree distributions across datasets to enable transfer between similar datasets due to its unique network structure, achieving strong scalability. In the following sections, we will provide detailed descriptions of these components.

4.1 The Training Phase

4.1.1 Forward and Recovery. Graph diffusion models inherently involve high complexity in both time and space, posing significant

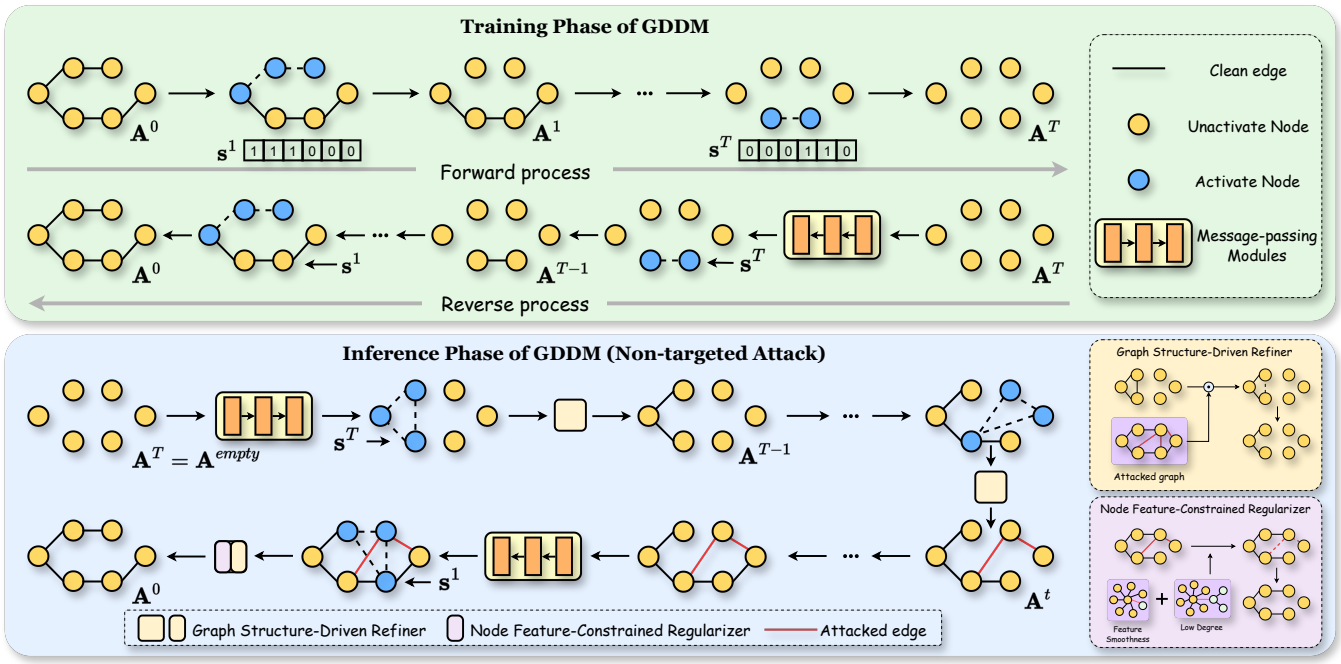


Figure 2: The Framework of GDDM.

challenges for efficient computation. To address this challenge while preserving the model's ability to accurately reconstruct graph structures, we simplify the model by introducing a state vector $\mathbf{s}^t$ [2], which tracks the nodes with degree changes at each timestep t of the discrete graph diffusion process. This simplification reduces computational overhead while maintaining the model's ability to restructure graphs effectively. Specifically, the state vector $\mathbf{s}^t$ indicates whether the degree of node i changes between timestep $t-1$ and t. If the degree changes, $\mathbf{s}_i^t$ is set to 1, otherwise 0. The **forward diffusion process** of GDDM can be formalized as:

$$q(\mathbf{A}^t, \mathbf{s}^t | \mathbf{A}^{t-1}) = q(\mathbf{A}^t | \mathbf{A}^{t-1}) = \mathcal{B}(\mathbf{A}^t; (1-\beta_t)\mathbf{A}^{t-1} + \beta_t p). \quad (4)$$

where β_t controls the probability of resampling edges. Notably, the introduction of $\mathbf{s}^t$ does not modify the core diffusion process but provides a mechanism to track which nodes' degrees are altered. Furthermore, setting $p = 0$ transforms the forward diffusion into an edge removal process.

The **recovery process** aims to refine the perturbed graph by progressively recovering its original structure. During the recovery process, we train a message-passing neural network $p_\theta(\mathbf{A}^{t-1}|\mathbf{A}^t)$ to approximate the posterior of the forward process:

$$q(\mathbf{A}^{t-1}|\mathbf{A}^t, \mathbf{s}^t, \mathbf{A}^0) = \frac{q(\mathbf{A}^t|\mathbf{A}^{t-1})q(\mathbf{s}^t|\mathbf{A}^t, \mathbf{A}^{t-1})q(\mathbf{A}^{t-1}|\mathbf{A}^0)}{q(\mathbf{A}^t|\mathbf{A}^0)q(\mathbf{s}^t|\mathbf{A}^t, \mathbf{A}^0)}. \quad (5)$$

Further derivation details are provided in Appendix A.1. In each recovery step, the network computes the state vector $\mathbf{s}^t$, which identifies the nodes for which edges should be generated. The recovery process is decomposed as follows:

$$p_\theta(\mathbf{A}^{t-1}|\mathbf{A}^t) = p_\theta(\mathbf{A}^{t-1}|\mathbf{s}^t, \mathbf{A}^t)p_\theta(\mathbf{s}^t|\mathbf{A}^t). \quad (6)$$

Although this decomposition introduces new optimization objectives, previous work [37] has shown a strong correlation between the state vector $\mathbf{s}^t$ and the degree vectors $\mathbf{d}^0$ and $\mathbf{d}^t$. The state vector can be predicted as:

$$q(\mathbf{s}^t|\mathbf{d}^t, \mathbf{d}^0) = \prod_{i=1}^{N} q(\mathbf{s}_i^t|\mathbf{d}_i^t, \mathbf{d}_i^0), \quad (7)$$

$$q(\mathbf{s}_i^t|\mathbf{d}_i^t, \mathbf{d}_i^0) = \mathcal{B}(\mathbf{s}_i^t; 1 - (1 - \frac{\beta_t \bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}})^{\mathbf{d}_i^0 - \mathbf{d}_i^t}),$$

where N represents the total number of nodes in the graph, $\alpha_t = 1-\beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. During the training phase, the optimization is focused only on $p_\theta(\mathbf{A}^{t-1}|\mathbf{s}^t, \mathbf{A}^t)$. The state vector $\mathbf{s}^t$ is directly sampled from the degree vectors $\mathbf{d}^0$ and $\mathbf{d}^t$, which simplifies the training phase by eliminating the need for additional optimization of $p_\theta(\mathbf{s}^t|\mathbf{A}^t)$, thus improving computational efficiency without compromising model performance. The detailed derivation of Eq. 7 can be found in the Appendix A.2.

4.1.2 Optimization Objective. Building on the recovery process discussed earlier, we now define the optimization objective for GDDM. To optimize the parameters θ, we maximize the variational lower bound (VLB) [10, 29] of $\log p(\mathbf{A}^0)$, ensuring that the model learns to accurately reconstruct the original graph from its perturbed version. The optimization objective is defined as follows:

$$\mathcal{L} = \mathbb{E}_q \left[\log \frac{p(\mathbf{A}^T)}{q(\mathbf{A}^T|\mathbf{A}^0)} + \log p_\theta(\mathbf{A}^0|\mathbf{A}^1, \mathbf{s}^1) + \sum_{t=2}^{T} \log \frac{p_\theta(\mathbf{A}^{t-1}|\mathbf{A}^t, \mathbf{s}^t)}{q(\mathbf{A}^{t-1}|\mathbf{A}^t, \mathbf{A}^0, \mathbf{s}^t)} \right]. \quad (8)$$

The first term does not involve learnable parameters, while Monte Carlo estimation [9] is used to optimize the second and third terms. This process enables GDDM to iteratively reconstruct the graph from an empty state, capturing its structure and improving denoising performance. The overall training phase of GDDM is presented in Appendix A.3.

4.1.3 Model Structure. To better understand how GDDM functions, we now provide an overview of its internal structure. GDDM consists of two main components: information propagation and edge prediction, which work together to reconstruct the graph from an empty structure iteratively.

In the information propagation process, node degree representations are used to capture features. The initial representation of each node is constructed by concatenating the node's initial $\mathbf{d}^0$ with its current degree $\mathbf{d}^t$, along with the timestep t and global contextual features $\mathbf{c}$. This can be formulated as:

$$\mathbf{Z}_i^0 = \text{concat}(\text{emb}_d(\mathbf{d}_i^t/\mathbf{d}_{max}) \| \text{emb}_d(\mathbf{d}_i^0/\mathbf{d}_{max})), \tag{9}$$

$$\mathbf{t}_{emb} = \text{emb}_t(t), \mathbf{c}^0 = \text{mean}(\mathbf{Z}^0), \tag{10}$$

where $\text{concat}(\cdot)$ represents the concatenation operation, $\text{emb}_d(\cdot)$ and $\text{emb}_t(\cdot)$ represent the degree and time encoding functions, $\mathbf{c}^0$ captures the initial global context, and $\mathbf{d}_{max}$ represents the maximum node degree in the original graph. Then, these initial node representations are updated iteratively as follows:

$$\mathbf{Z}^l, \mathbf{c}^l, \mathbf{H}^l = \text{MPM}(\mathbf{Z}^{l-1}, \mathbf{t}_{emb}, \mathbf{c}^{l-1}, \mathbf{H}^{l-1}), \tag{11}$$

where $\mathbf{H}$ refers to the node hidden features at each layer l. The message-passing mechanism (MPM) propagates information across the graph to refine the node representations iteratively. The detail of the information propagation process is in Appendix A.4.

For edge prediction, after L iterations of updates, a multi-layer perceptron (MLP) is used to predict whether an edge should be generated between nodes i and j. The prediction is based on the final representations $\mathbf{Z}_i^L$ and $\mathbf{Z}_j^L$ of the nodes:

$$b_{i,j}^{t-1} = \text{MLP}(\mathbf{Z}_i^L + \mathbf{Z}_j^L), \text{where } (i,j) \in \{(i,j) | \mathbf{s}_{i,j}^t = 1\}, \tag{12}$$

where $b_{i,j}^{t-1}$ is the predicted two-dimensional Bernoulli parameter for generating an edge between node i and node j. Then, whether an edge is generated between nodes i and j is determined by a sampling process based on $b_{i,j}^{t-1}$ and a Gumbel noise [12] g with the same dimensionality as $b_{i,j}^{t-1}$, which can be defined as follows:

$$\mathbf{A}_{i,j}^{t-1} = \arg\max(b_{i,j}^{t-1} + g). \tag{13}$$

4.2 The Inference Phase

Following the completion of the training phase, GDDM applies the learned diffusion dynamics to reconstruct the clean graph from the attacked one. During inference, the process iteratively restores the graph by sampling the state vector $\mathbf{s}^t$ and adjacency matrix $\mathbf{A}^{t-1}$ at each timestep, as defined by:

$$\mathbf{s}^t \sim q(\mathbf{s}^t | \mathbf{d}^t, \mathbf{d}^0), \tag{14}$$

$$\mathbf{A}^{t-1} \sim p_\theta(\mathbf{A}^{t-1} | \mathbf{s}^t, \mathbf{A}^t), \tag{15}$$

where $\mathbf{d}^0$ is the degree vector of the attacked graph, $\mathbf{d}^t$ is the degree vector at timestep t, and p_θ represents the learnable network.

Following the sampling steps in inference, the next crucial task is to ensure that the generated graph maintains a high-fidelity with the attacked graph. This is essential to ensure that while adversarial noise is mitigated, the core structure of the original graph is preserved. To achieve this, GDDM integrates two core components: Graph Structure-Driven Refiner (GSDR) and Node Feature-Constrained Regularizer (NFCR). Additionally, tailored denoising strategies are employed to handle various types of adversarial attacks, allowing GDDM to effectively target both global and localized attacks.

4.2.1 Graph Structure-Driven Refiner (GSDR). GSDR is designed to tackle the challenge of generating high-fidelity graphs during the denoising process. Due to the complexity of graph topology and the inherent noise in the original graph, directly generating a high-fidelity graph is difficult. GSDR addresses this by incrementally refining the graph structure at each timestep. At each step of the generation process, GSDR filters out incorrect edges, ensuring that only edges consistent with the attacked graph are retained. This process is formalized as:

$$\mathbf{A}^t = \mathbf{A}' \odot \mathbf{A}^t, \tag{16}$$

where $\mathbf{A}'$ represents the adjacency matrix of the attacked graph, $\mathbf{A}^t$ represents the adjacency matrix of the corresponding graph at timestep t during the generation process. The element-wise multiplication $\odot$ preserves the intersection between $\mathbf{A}'$ and $\mathbf{A}^t$. By filtering edges step-by-step, GSDR preserves plausible connections while removing adversarial noise, ensuring the generated graph remains structurally consistent with the attacked graph.

4.2.2 Node Feature-Constrained Regularizer (NFCR). While GSDR effectively removes the most noisy edges, residual noise in node features can still affect the overall graph quality. NFCR refines the generated graph by incorporating node feature and degree information, which addresses residual noise that remains after structural pruning. NFCR ensures that both node features and the underlying graph structure are preserved, enhancing the fidelity of the generated graph. NFCR operates in two stages: feature smoothness guiding and node degree guiding.

Feature smoothness guiding. As the generated graph reaches a certain size, it becomes necessary to evaluate whether the generated edges connect nodes that should logically be connected. This is achieved by assessing the feature smoothness between pairs of nodes connected by the generated edges. The feature smoothness is calculated as follows:

$$\mathbf{FS}(i,j) = \|\mathbf{X}_i - \mathbf{X}_j\|_2^2, \tag{17}$$

where $\mathbf{X}$ is the feature matrix of the original graph. Each row vector $\mathbf{X}_i$ or $\mathbf{X}_j$ in the feature matrix $\mathbf{X}$ represents the one-hot vector of a node, $\|\cdot\|^2$ represents the square of the second-order norm of the vector. $\mathbf{FS}(i,j)$ represents the feature smoothness between the two nodes, with higher values indicating lower feature similarity and a reduced likelihood of an edge existing between them. After calculating the feature smoothness for all edges in the generated graph, the top-k edges with the highest smoothness values (i.e., lowest feature similarity) are removed. This step ensures that only edges connecting nodes with similar features are retained, improving the quality of the generated graph.

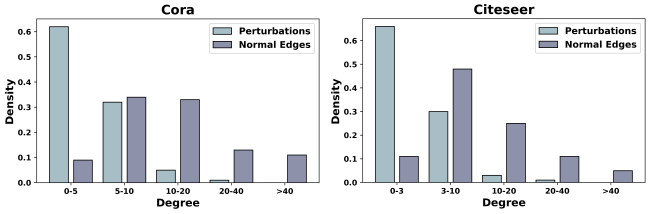


Figure 3: The edge degree distribution of the clean graph and the distribution of adversarial edges generated by GraD on the Cora and Citeseer dataset.

Node degree guiding. In addition to feature smoothness, the node degrees of the connected nodes also play a crucial role in determining whether an edge should be retained. Previous studies, such as GraD [24], reveal that graph attack methods tend to target low-degree nodes by introducing noisy edges that connect these vulnerable nodes. To explore this, we visualize the distribution of edges in the attacked graph, as shown in Fig. 3. In the visualization, green bars show the proportion of normal edges in each degree range relative to all normal edges, while orange bars indicate the proportion of attacked edges in each degree range relative to all attacked edges. We define an edge's degree as the sum of the degrees of its two connected nodes. As shown in the figure, attack methods frequently add edges that involve two low-degree nodes. This pattern indicates that low-degree nodes are more susceptible to being targeted by these attacks.

To address this vulnerability, the node degree information is integrated into the edge filtering process. Specifically, if an edge exhibits high feature smoothness (indicating a low likelihood of existence) and involves low-degree nodes, it is likely to be a noisy edge introduced by the attack. The probability of an edge being targeted by an attack is defined as:

$$P_{\text{attack}}(i,j) = \begin{cases} 1 & \text{if } d(i) < \lambda \text{ or } d(j) < \lambda \\ 0 & \text{else} \end{cases}, \tag{18}$$

where $d(\cdot)$ represents the degree of the target node in the attacked graph. λ represents the threshold, which varies depending on the dataset. After an edge is flagged based on node degrees, it undergoes further evaluation using feature smoothness. If $P_{\text{attack}}(i,j) = 1$ and the edge rank within the top-k in terms of feature smoothness, the edge is removed from the generated graph. Otherwise, the edge is retained.

By combining feature smoothness and node degree filtering, NFCR effectively removes noisy edges, especially those from low-degree nodes targeted by adversarial attacks. Together with GSDR, which ensures structural consistency by pruning incorrect edges, these components work to maintain the fidelity of the generated graph. While GSDR focuses on preserving the correct structure, NFCR refines node-level details, ensuring a clean and accurate graph, addressing **Challenge #1: Graph Fidelity**.

4.2.3 Tailored Attack-Specific Denoising Strategies (TADS). While GSDR and NFCR address the global fidelity of the graph, targeted attacks require more localized denoising strategies. To address **Challenge #2: localized denoising**, GDDM adopts tailored strategies for different types of adversarial attacks. For **non-targeted attacks**, the adjacency matrix $\mathbf{A}^{empty}$ of an empty graph is used as the starting point, allowing GDDM to reconstruct the entire graph from scratch. This global denoising process ensures that all edges are regenerated, as shown in Fig. 2. For **targeted attacks**, where specific nodes are compromised, a more focused strategy is necessary. GDDM starts by removing all edges connected to the target nodes, constructing an initial adjacency matrix $\mathbf{A}^{tar}$, defined as:

$$\mathbf{A}^{tar} = \mathbf{A}' - \mathbf{A}'', \tag{19}$$

where $\mathbf{A}'$ represents the adjacency matrix of the attacked graph, $\mathbf{A}''$ represents the adjacency matrix containing only the edges connected to the target nodes. This allows GDDM to perform localized denoising, focusing on the areas most affected by the attack, as shown in Appendix A.5.

4.3 Inherent Strong Scalability

The impractically high training cost of graph diffusion models on large datasets is a key challenge that hinders their application in graph purification. This limitation (**Challenge #3: Scalability**) prevents these models from scaling to larger datasets within a short period. Unlike existing graph diffusion models, GDDM does not rely on the specific feature dimensions of nodes in a graph dataset. Instead, it utilizes node degree information as the initial node features, allowing for easier transfer from smaller to larger datasets for denoising. This is facilitated by the structural similarity in degree distributions across different graph datasets (e.g., citation networks), reducing the need for retraining. Specifically, when defending against adversarial attacks on larger datasets, the model trained on a smaller dataset can be directly used, with only the node features being reinitialized based on the degree distribution of the new graph dataset. We present detailed experimental results in Section 5.2 and demonstrate that the scalability of GDDM can be successfully applied across similar datasets.

5 Experiment

In this section, we present our experimental setup and the empirical evaluation of GDDM's defense effectiveness and robustness against different adversarial attacks.

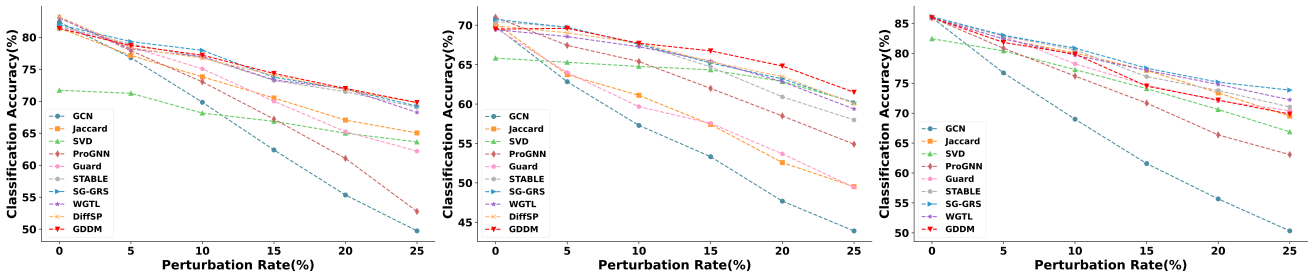
5.1 Experimental Settings

5.1.1 Dataset. We conduct our experiments on three datasets , consisting of Cora and Citeseer, and Pubmed. Notably, we directly apply the training parameters from the smaller Cora dataset to defend against adversarial attacks on Pubmed dataset, thereby verifying the scalability of GDDM. For each graph, we randomly split the nodes into 10% for training, 10% for validation, and 80% for testing. The statistics for these datasets are provided in Appendix B.1.

5.1.2 Experimental Details. We compare GDDM with representative and state-of-the-art defense methods including classical GNNs method [18], improved GCNs methods [1, 11, 15, 22], graph purification methods [4, 21, 36] and graph diffusion-based method [25] to evaluate its robustness. We mainly consider **poisoning attacks**(non-targeted structure attack [24] and targeted structure attack [46]) in this work. Our experiments follow the **transductive setting**, where the training and test nodes are available during training—an

Table 1: Node classification performance (Accuracy±Std) under targeted attack (Nettack). Results marked with * are obtained by transferring parameters from Cora dataset

Datasets	Ptb Num	GCN	Jaccard	SVD	ProGNN	Guard	STABLE	SG-GRS	WGTL	DiffSP	GDDM
Cora	0	81.46±1.04	78.88±0.67	77.64±2.48	**83.26±0.79**	81.35±1.15	81.01±0.98	82.23±0.93	81.80±1.27	82.14±0.84	82.02±1.00
	1	75.51±1.31	74.27±0.93	78.09±1.61	77.53±1.59	78.88±1.57	78.31±1.50	80.12±1.33	79.37±1.17	80.47±0.98	**81.34±1.43**
	2	71.01±1.57	70.11±1.25	76.97±1.15	74.16±1.00	75.84±1.35	76.52±0.98	78.78±0.90	78.12±1.09	79.44±1.15	**81.46±1.03**
	3	67.87±1.76	68.09±1.03	71.46±2.02	67.53±0.79	69.21±1.35	73.60±0.95	77.02±1.06	76.83±1.35	77.61±1.04	**80.78±0.93**
	4	63.93±1.98	64.94±1.73	70.45±3.30	65.06±2.38	65.96±0.88	72.69±0.93	75.31±1.27	75.03±0.97	75.93±1.17	**79.77±1.01**
	5	57.64±1.88	63.15±1.73	64.04±3.41	57.19±2.63	61.91±1.46	68.65±1.35	72.17±1.58	72.66±1.18	74.49±0.97	**78.65±1.74**
	Mean	69.72	69.91	73.11	70.79	72.19	75.13	77.61	77.30	78.35	**80.67**
Citeseer	0	83.12±1.12	82.71±0.83	78.02±1.35	83.23±0.96	83.54±0.78	85.00±1.91	**85.35±2.20**	84.56±1.23	84.74±0.99	83.03±1.05
	1	76.25±1.38	77.50±2.60	78.12±0.81	81.25±1.80	79.69±1.82	**84.47±1.73**	82.66±1.64	82.38±1.07	82.51±1.38	82.50±1.21
	2	60.10±4.52	65.62±2.13	69.90±2.35	69.68±5.86	74.17±0.78	70.83±4.42	81.15±2.51	80.84±1.55	81.33±1.01	**81.25±1.47**
	3	54.48±1.92	64.79±3.01	70.00±1.21	58.64±2.47	73.65±1.68	64.69±3.42	80.59±1.28	77.33±1.34	80.36±1.23	**81.87±1.06**
	4	48.44±2.43	58.12±1.67	62.08±3.13	53.23±1.35	71.77±2.16	61.04±2.21	79.28±1.80	76.14±1.28	79.67±1.17	**82.60±1.61**
	5	43.33±2.95	52.50±1.99	57.40±4.26	47.50±4.57	63.44±3.00	56.04±4.57	78.19±1.74	74.87±0.88	78.51±1.39	**81.46±1.02**
	Mean	60.95	66.87	69.25	65.59	74.27	70.35	81.20	79.35	81.19	**82.12**
Pubmed	0	89.12±0.36	87.72±0.74	84.23±0.89	89.37±0.74	89.14±0.67	89.04±0.47	**89.19±0.83**	89.07±0.82	OOM	89.13±0.66*
	1	87.73±0.47	86.88±0.51	83.25±0.61	87.21±0.83	87.96±0.53	87.13±0.62	88.33±1.14	86.69±0.61	OOM	**88.45±0.78***
	2	83.87±0.61	86.06±0.69	82.18±0.78	85.61±0.93	87.24±0.77	86.48±0.57	87.91±1.30	85.73±0.92	OOM	**88.01±0.79***
	3	80.14±0.58	85.03±0.37	81.09±0.84	84.23±1.14	86.39±0.71	85.51±0.61	87.20±1.49	83.12±0.88	OOM	**87.68±0.51***
	4	75.39±0.75	83.94±0.56	80.23±0.67	80.14±0.83	85.64±0.46	84.87±0.48	86.53±0.91	81.71±0.87	OOM	**87.23±0.81***
	5	65.81±0.47	83.53±0.77	77.35±1.02	71.86±0.91	85.06±0.78	84.26±0.59	86.11±1.32	80.48±0.81	OOM	**86.94±0.67***
	Mean	80.34	85.53	81.39	83.07	86.91	86.22	87.55	84.47	OOM	**87.91***



(a) Cora (b) Citeseer (c) Pubmed

Figure 4: Node classification performance (Accuracy±Std) under non-targeted attack (GraD).

approach widely adopted in adversarial defense for node classification. We provide descriptions of the baselines, implementation details in Appendix B.2 and Appendix B.3, respectively.

5.2 Robustness Against Poisoning Attacks

5.2.1 Performance Against Targeted Attack. **Q: Does GDDM outperform state-of-the-art graph defense models across targeted attacks while maintaining strong scalability?** The answer is **Yes**, since GDDM outperforms other methods on the majority of datasets and can successfully transfer from small to large datasets for effective defense. Table 1 presents the performance on three datasets against targeted attack (Nettack [46]). We set the perturbation number for the targeted node between 1 and 5 with the step of 1. Following prior work [15], we selected nodes with degrees larger than 10 in the test set as target nodes for the targeted attacks. The top two results are highlighted in bold and underlined. From this main experiment, we make the following observations:

- The performance of all the methods declines as the perturbation number increases, demonstrating the effectiveness of the attack method. In most cases, GCN shows the largest performance drop, proving that its ability to defend against adversarial attacks is particularly weak.
- Jaccard and SVD are feature-based edge pruning methods that show lower sensitivity to perturbations compared to other baselines. However, they often fail to fully remove adversarial edges and may mistakenly delete clean edges, resulting in performance degradation on clean graphs. Guard uses a scoring function based on gradients and node degrees, therefore outperforming them in most cases.
- The improved GCN methods ProGNN, STABLE, SG-GRS and WGTL perform well with low perturbations but ProGNN

Table 2: Results of denoising strategies for a targeted attack.

Datasets	Ptb Num	GDDM w/o TADS	GDDM
Cora	**0**	80.80±1.57	**82.02±1.00**
	1	77.41±0.85	**81.34±1.43**
	2	72.58±1.35	**81.46±1.03**
	3	70.79±1.23	**80.78±0.93**
	4	66.06±1.87	**79.77±1.01**
	5	62.13±2.71	**78.65±1.74**
Citeseer	**0**	82.02±1.48	**83.03±1.05**
	1	81.25±1.55	**82.50±1.21**
	2	78.75±1.87	**81.25±1.47**
	3	78.02±1.51	**81.87±1.06**
	4	76.88±2.02	**81.60±1.61**
	5	79.06±1.58	**81.46±1.02**
Pubmed	**0**	86.78±0.62	**89.13±0.66**
	1	85.63±0.81	**88.45±0.78**
	2	84.43±0.72	**88.01±0.79**
	3	82.13±0.65	**87.68±0.51**
	4	80.07±0.74	**87.23±0.81**
	5	77.42±0.89	**86.94±0.67**

and STABLE suffer a sharp decline as perturbations increase, likely due to unreliable structure learning in heavily contaminated graphs. STABLE outperforms ProGNN by adding a greater number of reliable edges. SG-GRS and WGTL enhance their robustness under strong adversarial attacks by incorporating additional constraints.

- Graph diffusion-based model DiffSP models both the attack and defense processes using a diffusion-based framework, which contributes to its strong robustness on most datasets. However, despite its effectiveness, DiffSP suffers from poor scalability and fails to run on larger datasets like Pubmed due to high training cost.
- Experimental results on the Pubmed dataset show that GDDM effectively defends against adversarial attacks when transferred from a small to a large dataset, highlighting its strong scalability and providing a viable approach for deploying graph diffusion models on large-scale datasets.
- GDDM surpasses baseline methods across most perturbation levels, benefiting from the diffusion model's strong recovery capability in highly disruptive scenarios, along with our tailored denoising strategies and components that leverage structural and feature information for guidance.

5.2.2 Performance Against Non-targeted Attack. **Q: Does GDDM outperform state-of-the-art graph defense models across non-targeted attacks?** The answer is **Yes**, since GDDM performs well on most datasets in non-targeted attack scenarios. The performance under non-targeted attack is shown in Fig. 4. We chose the powerful attack method GraD which is an improved version of Metattack [47] as an attacker to perturb graph structure. As we can see, GDDM is competitive compared to Sota methods, and in most cases, its performance outperforms other baseline methods. Together with the observations from Section 5.2.1, we can conclude that GDDM is capable of effectively defending against both targeted and non-targeted attacks.

5.3 Impact of Attack-specific Denoising

Q: Do our designed denoising strategies enhance GDDM's ability to protect specific nodes under targeted attack? The answer is **Yes**, since the Tailored Attack-specific Denoising Strategies (TADS) enhances GDDM's defense effectiveness in targeted attack scenarios. To validate the effectiveness of our proposed TADS for targeted attacks, we conducted separate experiments in a targeted attack scenario using both GDDM's denoising strategies designed for targeted attacks and non-targeted attacks. The experimental results are shown in Table 2, where GDDM represents the performance of GDDM using the denoising strategy designed for targeted attacks in a targeted attack scenario, and GDDM w/o TADS represents the performance of GDDM using the denoising strategy for non-targeted attacks in the same targeted attack scenario. The results indicate that the performance of GDDM in the targeted attack scenario significantly improves when using the TADS, demonstrating that GDDM uses TADS to achieve targeted protection by focusing on the attacked nodes.

5.4 Ablation Study

Q: Do the additional components improve the model's defense effectiveness against various attack methods? The answer is **Yes**, the removal of different modules negatively impacts GDDM to varying degrees. To gain a deeper understanding of how different components impact the model's performance, we conduct an ablation study to evaluate the contributions of the various components in GDDM:

- **GDDM w/o NFCR**: the proposed GDDM without the Node Feature-Constrained Regularizer.
- **GDDM w/o GSDR**: the proposed GDDM without the Graph Structure-Driven Refiner.

We use the hyper-parameters that achieve the best performance in our main experiment and report the results of 10 runs in Table 3. By comparing the full GDDM model with its ablation versions **GDDM w/o GSDR**, **GDDM w/o NFCR**, we observe that the Graph Structure-Driven Refiner plays a crucial role in GDDM. This component guides the generation process using the structure of the attacked graph, ensuring the basic graph fidelity of the denoised graph. Furthermore, we also observe that both feature smoothness guiding and node degree guiding consist in the Node Feature-Constrained Regularizer future enhance the defense performance. The above two experimental phenomena highlight the importance of graph fidelity in defending against adversarial attacks in graph diffusion models.

5.5 Model Analyses

5.5.1 Parameter Analysis. **Q: How does the performance of GDDM vary with different hyper-parameters?** Drawing from our experimental observations, the graph size ratio μ plays a critical role in optimizing performance. Accordingly, we systematically vary its value to evaluate its influence on the performance of GDDM. The results are shown in Fig. 5. Drawing from our experimental observations, the graph size ratio μ plays a critical role in optimizing performance. Accordingly, we systematically vary its value to evaluate its influence on the performance of GDDM. It is worth noting that as the graph size ratio μ increases, the model's performance

Table 3: Ablation experiments under non-targeted attack (GraD) on Cora and Citeseer.

Dataset	Cora					Citeseer				
Ptb Rate	**5%**	**10%**	**15%**	**20%**	**25%**	**5%**	**10%**	**15%**	**20%**	**25%**
GDDM w/o GSDR	32.49±0.47	32.49±0.47	32.49±0.47	32.49±0.47	32.49±0.47	33.19±0.62	33.19±0.62	33.19±0.62	33.19±0.62	33.19±0.62
GDDM w/o NFCR	77.76±0.45	75.65±0.47	72.51±0.32	69.77±0.50	66.85±0.28	68.75±0.29	66.76±0.52	65.67±0.47	64.01±0.72	60.68±0.38
GDDM	**78.72±0.54**	**77.20±0.34**	**73.35±0.40**	**72.00±0.50**	**69.80±0.32**	**69.60±0.35**	**67.71±0.43**	**66.74±0.40**	**64.82±0.59**	**61.49±0.47**
Ptb Num	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
GDDM w/o GSDR	42.41±3.01	42.41±3.01	42.41±3.01	42.41±3.01	42.41±3.01	56.90±4.37	56.90±4.37	56.90±4.37	56.90±4.37	56.90±4.37
GDDM w/o NFCR	81.12±1.21	81.01±1.06	80.56±1.01	79.55±1.40	78.09±1.44	81.45±1.30	81.15±1.43	81.25±1.04	82.08±1.38	81.04±1.12
GDDM	**81.34±1.43**	**81.46±1.03**	**80.78±0.93**	**79.77±1.01**	**78.65±1.74**	**82.50±1.21**	**81.25±1.47**	**81.87±1.06**	**82.60±1.61**	**81.46±1.02**

Table 4: Average training time per epoch (sec/epoch) on the Cora dataset.

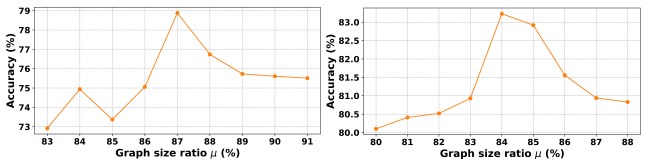
Model	SVD	Jaccard	Guard	ProGNN	STABLE	WGTL	SG-GSR	DiffSP	GDDM
sec/epoch	0.013	0.009	0.005	3.57	0.078	3.12	0.057	23.64	2.45

first improves and then declines. This may be because a graph size ratio that is too small can cause the model to remove clean edges from the attacked graph, while a ratio that is too large may lead to insufficient removal of noisy edges. Therefore, we select an optimal graph size ratio to achieve the best model performance.

5.5.2 Node Degree Distribution Similarity Analysis. **Q: Does node degree distribution similarity significantly affect the transfer performance of GDDM?** The answer is **Yes**. To investigate the impact of node degree distribution similarity on the transfer performance of GDDM, we trained the model on the QM9 dataset, which has low degree distribution similarity with Pubmed, and then transferred it to the Pubmed dataset. The results are shown in Table 5. We observed that the model trained on QM9 performed significantly worse on Pubmed compared to the model transferred from Cora. This result indicates that the similarity in node degree distributions plays an important role in the transferability of GDDM.

5.5.3 Complexity Analysis. **Q: Does GDDM have lower time complexity than existing graph diffusion models?** The answer is **Yes**. Let M denote the number of edges and K the number of nodes with changing degrees during the reverse process. At each step t, the MPNN requires $O(M)$ for node representations, $O(N)$ for predicting $\mathbf{s}^t$, and $O(K^2)$ for link predictions among K nodes. With $K \ll N$ in most cases, and T diffusion steps, the total complexity is $O(T \cdot \max(K^2, M))$. In contrast, prior models require $O(T \cdot N^2)$ for $O(N^2)$ link predictions per step. Additionally, since our model does not require retraining on large datasets, its time overhead in real-world scenarios is lower than other graph diffusion models.

To further validate the complexity advantage of GDDM over existing graph diffusion models, we compare the per-epoch training time of all baseline methods on the Cora dataset. The results are shown in Table 4. All methods were trained on the same data splits to ensure comparable training sizes across models. As shown in Table 4, GDDM incurs a higher per-epoch training cost (2.45 sec/epoch). However, its ability to transfer across datasets—such as from Cora to Pubmed—without retraining helps mitigate this overhead, making it a practically scalable solution.



(a) Cora (b) Citeseer

Figure 5: Graph size ratio parameter analysis.

Table 5: Impact of node degree similarity on GDDM.

Model	Source→Target	Accuracy (%)
GDDM	Cora→Pubmed	**86.94±0.67**
GDDM	QM9→Pubmed	69.73±1.51

6 Conclusion

Graph Neural Networks (GNNs) are highly sensitive to adversarial attacks on graph structures. To effectively defend against different types of adversarial attacks that can poison GNNs, we propose a novel graph purification method GDDM in this work. This method leverages the powerful denoising capabilities of diffusion models, supplemented by multiple key components and tailored denoising strategies to purify attacked graphs, effectively reducing the impact of various adversarial attacks on GNNs. Additionally, this method can leverage the similarity in node degree distributions across graph datasets to enable seamless transfer between similar datasets without retraining, achieving strong scalability. Our experiments demonstrate that GDDM is highly competitive in different types of attack scenarios.

7 Acknowledgments

This work was supported by a grant from the National Natural Science Foundation of China under grants (No.62372211), and the Science and Technology Development Program of Jilin Province (No. 20220201153GX).

References

[1] Naheed Anjum Arafat, Debabrota Basu, Yulia Gel, and Yuzhou Chen. 2025. When witnesses defend: A witness graph topological layer for adversarial graph learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 39. 15408–15416.

[2] Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. 2023. Efficient and degree-guided graph generation via discrete diffusion modeling. *arXiv preprint arXiv:2305.04111* (2023).

[3] Tyler Derr, Yao Ma, Wenqi Fan, Xiaorui Liu, Charu Aggarwal, and Jiliang Tang. 2020. Epidemic graph convolutional network. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 160–168.

[4] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. 2020. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th international conference on web search and data mining*. 169–177.

[5] Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. 2022. Graph trend filtering networks for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 112–121.

[6] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

[7] Sicheng Gao, Xuhui Liu, Bohan Zeng, Sheng Xu, Yanjing Li, Xiaoyan Luo, Jianzhuang Liu, Xiantong Zhen, and Baochang Zhang. 2023. Implicit diffusion models for continuous super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10021–10030.

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

[9] John Hammersley. 2013. *Monte carlo methods*. Springer Science & Business Media.

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

[11] Yeonjun In, Kanghoon Yoon, Kibum Kim, Kijung Shin, and Chanyoung Park. 2024. Self-Guided Robust Graph Structure Refinement. In *Proceedings of the ACM on Web Conference 2024*. 697–708.

[12] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* (2016).

[13] Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. 2021. Node similarity preserving graph convolutional networks. In *Proceedings of the 14th ACM international conference on web search and data mining*. 148–156.

[14] Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. 2021. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter* 22, 2 (2021), 19–34.

[15] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 66–74.

[16] Wei Jin, Tong Zhao, Jiayuan Ding, Yozen Liu, Jiliang Tang, and Neil Shah. 2023. Empowering Graph Representation Learning with Test-Time Graph Transformation. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=Lnxl5pr018

[17] Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. 2022. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International conference on machine learning*. PMLR, 10362–10383.

[18] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[19] Lingkai Kong, Jiaming Cui, Haotian Sun, Yuchen Zhuang, B Aditya Prakash, and Chao Zhang. 2023. Autoregressive diffusion model for graph generation. In *International conference on machine learning*. PMLR, 17391–17408.

[20] Harjinder Singh Lallie, Kurt Debattista, and Jay Bal. 2020. A review of attack graph and attack tree visual syntax in cyber security. *Computer Science Review* 35 (2020), 100219.

[21] Jintang Li, Jie Liao, Ruofan Wu, Liang Chen, Zibin Zheng, Jiawang Dan, Changhua Meng, and Weiqiang Wang. 2023. GUARD: Graph universal adversarial defense. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1198–1207.

[22] Kuan Li, Yang Liu, Xiang Ao, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. Reliable representations make a stronger defender: Unsupervised structure refinement for robust gnn. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 925–935.

[23] Gang Liu, Eric Inae, Tong Zhao, Jiaxin Xu, Tengfei Luo, and Meng Jiang. 2024. Data-centric learning from unlabeled graphs with diffusion model. *Advances in neural information processing systems* 36 (2024).

[24] Zihan Liu, Yun Luo, Lirong Wu, Zicheng Liu, and Stan Z Li. 2023. Towards reasonable budget allocation in untargeted graph structure attacks via gradient debias. *arXiv preprint arXiv:2304.00010* (2023).

[25] Jiayi Luo, Qingyun Sun, Haonan Yuan, Xingcheng Fu, and Jianxin Li. 2025. Robust Graph Learning Against Adversarial Evasion Attacks via Prior-Free Diffusion-Based Structure Purification. In *Proceedings of the ACM on Web Conference 2025*. 2098–2110.

[26] Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. 2022. Are defenses for graph neural networks robust? *Advances in Neural Information Processing Systems* 35 (2022), 8954–8968.

[27] William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4195–4205.

[28] Tahereh Pourhabibi, Kok-Leong Ong, Booi H Kam, and Yee Ling Boo. 2020. Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems* 133 (2020), 113303.

[29] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*. PMLR, 2256–2265.

[30] Lichao Sun, Yingtong Dou, Carl Yang, Kai Zhang, Ji Wang, S Yu Philip, Lifang He, and Bo Li. 2022. Adversarial attack and defense on graph data: A survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 8 (2022), 7693–7711.

[31] Zhiqing Sun and Yiming Yang. 2023. Difusco: Graph-based diffusion solvers for combinatorial optimization. *Advances in Neural Information Processing Systems* 36 (2023), 3706–3731.

[32] Curtis R Taylor, Krishna Venkatasubramanian, and Craig A Shue. 2014. Understanding the security of interoperable medical devices using attack graphs. In *Proceedings of the 3rd international conference on High confidence networked systems*. 31–40.

[33] Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. 2022. Digress: Discrete denoising diffusion for graph generation. *arXiv preprint arXiv:2209.14734* (2022).

[34] Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. 2024. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8228–8238.

[35] Yujie Wang, Shuo Zhang, Junda Ye, Hao Peng, and Li Sun. 2024. A Mixed-Curvature Graph Diffusion Model. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2482–2492.

[36] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv preprint arXiv:1903.01610* (2019).

[37] Mingyang Wu, Xiaohui Chen, and Liping Liu. 2023. EDGE++: Improved Training and Sampling of EDGE. *arXiv preprint arXiv:2310.14441* (2023).

[38] Bin Xia, Yulun Zhang, Shiyin Wang, Yitong Wang, Xinglong Wu, Yapeng Tian, Wenming Yang, and Luc Van Gool. 2023. Diffir: Efficient diffusion model for image restoration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 13095–13105.

[39] Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. 2021. Graph-revised convolutional network. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part III*. Springer, 378–393.

[40] Xiangchi Yuan, Chunhui Zhang, Yijun Tian, Yanfang Ye, and Chuxu Zhang. 2024. Mitigating emergent robustness degradation while scaling graph learning. The Twelfth International Conference on Learning Representations (ICLR).

[41] Chunhui Zhang, Yijun Tian, Mingxuan Ju, Zheyuan Liu, Yanfang Ye, Nitesh Chawla, and Chuxu Zhang. 2022. Chasing all-round graph representation robustness: Model, training, and optimization. In *The eleventh international conference on learning representations*.

[42] Xiang Zhang and Marinka Zitnik. 2020. Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in neural information processing systems* 33 (2020), 9263–9275.

[43] Qinkai Zheng, Xu Zou, Yuxiao Dong, Yukuo Cen, Da Yin, Jiarong Xu, Yang Yang, and Jie Tang. 2021. Graph robustness benchmark: Benchmarking the adversarial robustness of graph machine learning. *arXiv preprint arXiv:2111.04314* (2021).

[44] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1399–1407.

[45] Xu Zou, Qinkai Zheng, Yuxiao Dong, Xinyu Guan, Evgeny Kharlamov, Jialiang Lu, and Jie Tang. 2021. Tdgia: Effective injection attacks on graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2461–2471.

[46] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2847–2856.

[47] Daniel Zügner and Stephan Günnemann. 2019. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *International Conference on Learning Representations (ICLR)*.

A Detailed Implementations

A.1 Derivation Process of the Posterior

The further derivation of Eq. 5 are shown as follows:

$$\begin{aligned} q(\mathbf{A}^{t-1}|\mathbf{A}^t,\mathbf{s}^t,\mathbf{A}^0) &= \frac{q(\mathbf{A}^{t-1},\mathbf{A}^t,\mathbf{s}^t|\mathbf{A}^0)}{q(\mathbf{A}^t,\mathbf{s}^t|\mathbf{A}^0)} \\ &= \frac{q(\mathbf{A}^t|\mathbf{A}^{t-1},\mathbf{A}^0)q(\mathbf{s}^t|\mathbf{A}^0,\mathbf{A}^t,\mathbf{A}^{t-1})q(\mathbf{A}^{t-1}|\mathbf{A}^0)}{q(\mathbf{A}^t,\mathbf{s}^t|\mathbf{A}^0)} \\ &= \frac{q(\mathbf{A}^t|\mathbf{A}^{t-1})q(\mathbf{s}^t|\mathbf{A}^t,\mathbf{A}^{t-1})q(\mathbf{A}^{t-1}|\mathbf{A}^0)}{q(\mathbf{A}^t|\mathbf{A}^0)q(\mathbf{s}^t|\mathbf{A}^t,\mathbf{A}^0)}. \end{aligned} \tag{20}$$

A.2 The Detailed Derivation of Eq. 7

The content of Eq. 7 is shown below:

$$q(\mathbf{s}^t|\mathbf{d}^t,\mathbf{d}^0) = \prod_{i=1}^{N} q(\mathbf{s}_i^t|\mathbf{d}_i^t,\mathbf{d}_i^0), \tag{21}$$

$$q(\mathbf{s}_i^t|\mathbf{d}_i^t,\mathbf{d}_i^0) = \mathcal{B}(\mathbf{s}_i^t; 1-(1-\frac{\beta_t\bar{\alpha}_{t-1}}{1-\bar{\alpha}_{t-1}})^{\mathbf{d}_i^0-\mathbf{d}_i^t}),$$

For the second line of Equation 7, we have added the detailed derivation process as follows:

First, the degree distribution in the forward diffusion process of the graph takes the following form:

$$q(\mathbf{d}^t|\mathbf{d}^0) = \prod_{i=1}^{N} q(\mathbf{d}_i^t|\mathbf{d}_i^0), \tag{22}$$

$$\text{where } q(\mathbf{d}_i^t|\mathbf{d}_i^0) = \text{Binomial}(k=\mathbf{d}_i^t, n=\mathbf{d}_i^0, p=\bar{\alpha}_t).$$

$$q(\mathbf{d}^t|\mathbf{d}^{t-1}) = \prod_{i=1}^{N} q(\mathbf{d}_i^t|\mathbf{d}_i^{t-1}), \tag{23}$$

$$\text{where } q(\mathbf{d}_i^t|\mathbf{d}_i^{t-1}) = \text{Binomial}(k=\mathbf{d}_i^t, n=\mathbf{d}_i^{t-1}, p=1-\beta_t).$$

Intuitively, for $q(\mathbf{d}^t|\mathbf{d}^0)$, each of the $\mathbf{d}_i^0$ edges connected to node i has a probability of $\bar{\alpha}$ of being retained at time step t. Therefore, the probability that the number of remaining edges equals $\mathbf{d}_i^t$ at time step t follows a binomial distribution. A similar conclusion holds for $q(\mathbf{d}^t|\mathbf{d}^{t-1})$, except that the probability becomes $1-\beta_t$. Based on the above, we can now derive the posterior degree distribution in the reverse process, which is expressed as follows:

$$q(\mathbf{d}^{t-1}|\mathbf{d}^0,\mathbf{d}^t) = \prod_{i=1}^{N} q(\mathbf{d}_i^{t-1}|\mathbf{d}_i^t,\mathbf{d}_i^0), \tag{24}$$

$$\begin{aligned} \text{where } q(\mathbf{d}_i^{t-1}|\mathbf{d}_i^t,\mathbf{d}_i^0) = \text{Binomial}(&k=\mathbf{d}_i^{t-1}-\mathbf{d}_i^t, \\ &n=\mathbf{d}_i^0-\mathbf{d}_i^t, p=\frac{\beta_t\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}). \end{aligned}$$

When a node's degree does not change between time steps $t-1$ and t, we always have $\mathbf{d}_i^t = \mathbf{d}_i^{t-1}$. When $\mathbf{d}_i^t = \mathbf{d}_i^{t-1}$,the probabilities of these events can be calculated by querying the degree distributions $q(\mathbf{d}^t|\mathbf{d}^{t-1})$ or $q(\mathbf{d}^{t-1}|\mathbf{d}^t,\mathbf{d}^0)$ as follows:

$$q(\mathbf{d}^t|\mathbf{d}^{t-1}) = (1-\beta_t)^{\mathbf{d}_i^{t-1}} \tag{25}$$

$$q(\mathbf{d}^{t-1}|\mathbf{d}^0,\mathbf{d}^t) = (1-\frac{\beta_t\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})^{\mathbf{d}_i^0-\mathbf{d}_i^t} \tag{26}$$

Algorithm 1 The Training Phase of GDDM

Input: The clean graph adjacency matrix $\mathbf{A}^0$, diffusion schedule β, the learnable parameter θ.

1: **while** model convergence **do**
2: Sample $t \sim \mathcal{U}[1,T]$, and obtain $t-1$.
3: Draw $\mathbf{A}^{t-1} \sim q(\mathbf{A}^{t-1}|\mathbf{A}^0)$ and $\mathbf{A}^t \sim q(\mathbf{A}^t|\mathbf{A}^{t-1})$.
4: Obtain $\mathbf{s}^t$ with $\mathbf{A}^{t-1}$ and $\mathbf{A}^t$.
5: Optimize the parameter θ by minimizing $\log \frac{p_\theta(\mathbf{A}^{t-1}|\mathbf{A}^t,\mathbf{s}^t)}{q(\mathbf{A}^{t-1}|\mathbf{A}^t,\mathbf{A}^0,\mathbf{s}^t)}$.
6: **end while**

Let $\mathbf{s}_i^t$ be the random variable that node i has degree change at time step t. Below we show the forward and reverse degree change distributions:

At timestep t, the forward degree change distribution for node i given $\mathbf{d}_i^{t-1}$ is:

$$q(\mathbf{s}_i^t|\mathbf{d}_i^{t-1}) = \mathcal{B}(\mathbf{s}_i^t; 1-(1-\beta_t)^{\mathbf{d}_i^{t-1}}). \tag{27}$$

At timestep t, the reverse degree change distribution for node i given $\mathbf{d}_i^0, \mathbf{d}_t^0$ is:

$$q(\mathbf{s}_i^t|\mathbf{d}_i^t,\mathbf{d}_i^0) = \mathcal{B}(\mathbf{s}_i^t; 1-(1-\frac{\beta_t\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t})^{\mathbf{d}_i^0-\mathbf{d}_i^t}). \tag{28}$$

Table 6: Statistics of the experimental data.

Dataset	N	E	Classes	Features
Cora	2,708	5,278	7	1,433
Citeseer	3,327	4,552	6	3,703
Pubmed	19,717	44,324	3	500

A.3 Algorithm of Training Phase

Graph diffusion models inherently involve high complexity in both time and space, posing significant challenges for efficient computation. To address this challenge while preserving the model's ability to accurately reconstruct graph structures, we simplify the model by introducing a state vector $\mathbf{s}^t$, which tracks the nodes with degree changes at each timestep t of the discrete graph diffusion process. This simplification reduces computational overhead while maintaining the model's ability to restructure graphs effectively. The training phase of GDDM is shown in Algorithm 1.

A.4 Information Propagation Process Module

In the information propagation process, the node representation $\mathbf{Z}$ and the representation $\mathbf{t}_{emb}$ of the timestep t are concatenated as the input of GT or GNN. GRU adaptively retains important information inputting the output of GT and the nodes' hidden features $\mathbf{H}$:

$$\mathbf{Z}^l = \text{concat}(\mathbf{Z}^l \| \mathbf{t}_{emb}\mathbf{1}^T), \tag{29}$$

$$\mathbf{Z}^l = \text{GT}(\mathbf{Z}^l,\mathbf{A}^t) \text{ or } \text{GNN}(\mathbf{Z}^l,\mathbf{A}^t), \tag{30}$$

$$\mathbf{Z}^l, \mathbf{H}^{l+1} = \text{GRU}(\mathbf{Z}^l,\mathbf{H}^l), \tag{31}$$

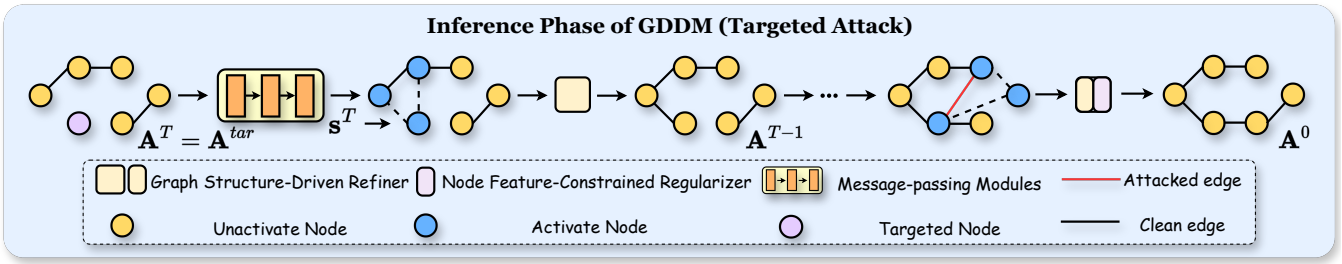


Figure 6: The Inference Phase of GDDM for Targeted Attacks.

where $\mathbf{t}_{emb}$ is the representation of timestep t. $\mathbf{A}^t$ is the adjacency matrix of the graph at the current timestep t. GT$(\cdot)$ and GNN$(\cdot)$ represent Graph Transformer and Graph Neural Network, which perform attentive information propagation between nodes, allowing each node to aggregate local feature information from its neighborhood in the graph. GRU$(\cdot)$ represents Gated Recurrent Unit (GRU).

To update the global contextual feature $\mathbf{c}^l$, the representation of each node and the current global contextual feature $\mathbf{c}^l$ are concatenated and applied a mean operation as the input of a MLP. The output of the MLP serves as the global contextual feature for the next layer:

$$\mathbf{c}^{l+1} = \text{MLP}(\text{mean}(\text{concat}(\mathbf{Z}^l \| \mathbf{c}^l \mathbf{1}^T))). \tag{32}$$

Finally, the node representations Z and the global contextual feature $\mathbf{c}^{l+1}$ are summed to obtain the initial node representations for the next MPM:

$$\mathbf{Z}^{l+1} = \mathbf{Z}^l + \mathbf{c}^{l+1} \mathbf{1}^T. \tag{33}$$

A.5 The Inference Phase of GDDM for Targeted Attack

GDDM adopts tailored strategies for different types of adversarial attacks. For **targeted attacks**, where specific nodes are compromised, a more focused strategy is necessary. GDDM starts by removing all edges connected to the target nodes, constructing an initial adjacency matrix $\mathbf{A}^{tar}$. The detail of the Tailored Attack-specific Denoising Strategies (TADS) is shown in Figure 6.

B Experiments Detail

B.1 Dataset

The statistics are listed in Tabel 6. Cora, Citeseer and Pubmed are citation networks where nodes represent papers from seven,six and three research categories, respectively, and edges denote citation links.

B.2 Baselines

- **GCN**: GCN is a widely used graph convolutional network grounded in spectral graph theory. It applies convolution operations on graphs to capture the relationships between nodes based on their connectivity.
- **Jaccard**: Jaccard prunes edges based on the feature similarity between nodes, as adversarial attacks tend to connect dissimilar nodes in the graph.
- **SVD**: SVD removes the high-rank perturbations introduced by adversarial attacks by the low-rank approximation of the attacked graph, enhancing the robustness of GNN model.
- **ProGNN**: ProGNN mitigates the negative impact of adversarial attacks on GNN models by optimizing the adjacency matrix of the attacked graph using regularization terms based on feature smoothness, low-rank, and sparsity.
- **Guard**: Guard uses a scoring function based on node degree and gradient to identify anchor nodes (attacker nodes), then removes all edges between the anchor nodes and the target node to ensure the target node is protected from attacks.
- **STABLE**: STABLE is an unsupervised method for optimizing graph structure, which leverages a specially designed robust GCN to effectively defend against adversarial attacks on graph data.
- **SG-GRS**: SG-GSR is a self-guided framework that enhances robustness against adversarial attacks by extracting a clean subgraph from the attacked graph itself.
- **WGTL**: WGTL improves GNN robustness by integrating local and global topological features through witness complexes and persistent homology. In this paper, we use GCN combined with WGTL as a baseline.
- **DiffSP**: DiffSP is a defense method that models both attack and purification through a unified graph diffusion process, enabling robust classification under structural perturbations.
- **GraD**: GraD introduces an improved attack objective function that enables the attack model to target medium-confidence nodes, thereby increasing attack efficiency.
- **Nettack**: Nettack is a greedy-based attack method that perturbs targeted nodes in a graph while preserving the node degree distribution and feature co-occurrence.

B.3 Implementation Details

We generate attacks on each graph based on the perturbation number/rate. During the model training phase, we primarily adjust the diffusion steps, with a range of $\{64, 128, 256, 512\}$. During the model inference phase, we primarily adjust the expected size of the generated graph, ranging from 70% to 95%. Performance is reported as the average accuracy with standard deviation, based on 10 runs on clean, perturbed or denoised graphs. We closely follow the default setting of the baselines for a fair comparison.