Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.076126

ARTICLE

Adversarial Attack Defense in Graph Neural Networks via Multiview Learning and Attention-Guided Topology Filtering

Cheng Yang, Xianghong Tang*, Jianguang Lu and Chaobin Wang

State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang, China

*Corresponding Author: Xianghong Tang. Email: xhtang@gzu.edu.cn

Received: 14 November 2025; Accepted: 09 February 2026; Published: 09 April 2026

ABSTRACT: Graph neural networks (GNNs) have demonstrated impressive capabilities in processing graph-structured data, yet their vulnerability to adversarial perturbations poses serious challenges to real-world applications. Existing defense methods often fail to handle diverse types of attacks and adapt to dynamic adversarial strategies because they typically rely on static defense mechanisms or focus narrowly on a single robustness dimension. To address these limitations, we propose an adversarial attention-based robustness strategy (AARS), which is a unified framework designed to enhance the robustness of GNNs against structural and feature perturbations. AARS operates in two stages: the first stage employs adversarial training with joint optimization to improve the resilience of the model to malicious attacks and stabilize its decision boundaries; the second stage incorporates an attention mechanism to identify critical structural dependencies and guides a topology filtering module that dynamically suppresses adversarial edges while preserving essential graph semantics. Extensive experiments on benchmark datasets for node classification demonstrate that AARS significantly outperforms existing baselines in terms of classification accuracy under various attack scenarios, thereby effectively improving the robustness of GNNs in static and dynamic adversarial settings.

KEYWORDS: Graph neural networks; adversarial robustness; adversarial training; attention mechanisms; adversarial attack defense

1 Introduction

Owing to its strong expressive power in modeling complex relational structures, graph-structured data has become increasingly important in applications such as social network analysis [1], biomedical sciences [2], and recommendation systems [3]. Unlike traditional structured data, graph-structured data feature complex topological relationships that capture higher-order dependencies among entities [4]. To effectively model these structures, graph neural networks (GNNs) have emerged as the preferred approach [5]. GNNs, which rely on a message-passing mechanism, iteratively gather neighborhood information to create node representations [6] and exhibit significant advantages in node classification tasks [7]. However, research indicates that GNNs are highly vulnerable to adversarial perturbations because attackers can significantly impair their classification performance by making small modifications to the graph structure or features, such as edge insertion/deletion or feature manipulation [8–11]. This weakness creates serious risks in security-critical areas such as medical diagnostics and financial risk management, where model reliability is essential. Therefore, improving the adversarial robustness of GNNs is crucial for ensuring dependable decision-making amid adversarial threats, and is vital for broadening their trustworthy use in real-world, high-stakes applications.

Various defense techniques have been studied to reduce the vulnerability of GNNs to adversarial attacks. Preprocessing-based methods focus on cleaning the input graph by removing suspicious edges or nodes, such as GCN-Jaccard [12], which eliminates edges with low node similarity, and GCNSVD [13], which performs low-rank approximation to decrease noise from perturbations. Graph structure learning (GSL) [14] methods dynamically refine the graph structure during training to enhance robustness, as exemplified by Pro-GNN [15], DefenseVGAE [16], and IDGL [17], which reconstruct cleaner or more resilient graph structures. Another approach modifies message passing mechanisms to reduce the adversarial influence propagation, including RGCN [18], which uses variance-aware attention; PA-GNN [19], which incorporates adversarial training and attention; and GNNGuard [20], which employs similarity-based attention with layer-wise memory. However, these methods typically rely on soft attention weighting, where all neighbors participate in message passing with varying importance scores, leaving the effective neighborhood size unconstrained and possibly worsening oversmoothing in multilayer propagation. Despite these advances, current defenses have clear limitations. Preprocessing methods depend on static assumptions and struggle with dynamic attacks. GSL approaches may increase graph connectivity or propagate messages over overly large neighborhoods, often leading to over-smoothing of representations and structural biases. Moreover, message-passing modifications are usually localized and inadequate for addressing diverse, adaptive attack strategies. These issues emphasize the need for more flexible and generalizable defense strategies to maintain robust GNN performance in complex adversarial environments.

To address these challenges, this study introduces an adversarial attention-based robustness strategy (AARS), a robust graph learning framework that uses adversarial training and a dynamic sparse attention mechanism (DSAM). AARS effectively reduces adversarial perturbations and simultaneously learns resilient graph structures through multiview adversarial training and adaptive topology refinement. Specifically, the framework addresses existing limitations in two main ways. First, it creates multiview adversarial samples to simulate various attack scenarios, helping the model learn perturbation-invariant representations during training. Second, it uses a DSAM with hard top-K neighbor selection based on node feature similarity, explicitly controlling the neighborhood size during message passing to prevent adversarial propagation and reduce over-smoothing. This dual approach maintains the balance between local smoothness and global structural integrity.

Our contributions can be summarized as follows.

- We develop a multiview adversarial training method that simulates various attack patterns, enabling the model to learn representations that are invariant to perturbations and thereby effectively improve its robustness against dynamic and evolving attacks.
- We present a DSAM that adaptively detects meaningful node connections based on feature similarity, reducing the spread of adversarial noise while preserving meaningful graph structures.
- Extensive experimental results demonstrate that AARS consistently enhances graph robustness and considerably improves the performance of GNNs on node classification tasks across various adversarial scenarios.

2 Related Works

Evolution of Adversarial Attacks on GNNs. GNNs, which depend on neighborhood structures and node features for message passing, have notable vulnerabilities in adversarial settings. Research indicates that even small changes to graph structure or features can significantly impair model performance. Structural attacks such as FGA [21] mislead the aggregation process by adding or removing critical edges. Feature attacks such as NIPA [22] induce misclassification by changing node attributes. Additionally, distributed attacks (e.g., Disttack) [23] and global attacks (e.g., Metattack) [24] reveal the transmissibility of perturbations

and their sensitivity to the training distribution [25]. As graph data become more dynamic, static attack assumptions can no longer fully address the complexity of how structures evolve and interact with disturbances in real-world situations. This study focuses on the vulnerability of GNNs when facing adversarial conditions, with the goal of enhancing model robustness and adaptability amid both structural changes and adversarial interference.

Advancements in GNN Defense Strategies. In response to the continuous evolution of attack techniques, defense strategies for GNNs have advanced from simple adversarial training to more sophisticated multimodal robustness optimization. Early methods [26,27] adapted adversarial paradigms from the image domain by introducing PGD-based structural perturbations for max-min robust optimization. However, these approaches often exhibited pseudo-robustness in transductive learning settings. To overcome these limitations, later studies proposed architectural improvements and regularization strategies [28] such as degree-based dynamic regularization, latent space perturbation generation, and smooth adversarial training, which strengthened the robustness of the model against both structural and feature perturbations while offering improved theoretical interpretability. To better balance the simulation of diverse attacks while maintaining structural semantics, we introduce a robust graph structure learning framework that combines adversarial training with attention mechanisms.

GNN Defense in Industrial IoT. GNNs have been increasingly applied to industrial IoT (IIoT) systems for tasks such as fault diagnosis, anomaly detection, and industrial network monitoring [29]. In such environments, attackers often operate in a black-box manner owing to limited system access, heterogeneous devices, and strict security constraints, making gradient-based adversarial attacks and defenses less practical. Recent studies have highlighted that robustness against data corruption and topology perturbations is critical in IIoT scenarios [30]. Therefore, defense strategies that do not rely on explicit gradient-based inner maximization, but instead enhance robustness under stochastic or distributional perturbations, are specifically appealing for industrial applications. Although our method is not specifically designed for IIoT systems, its randomized and gradient-free perturbation mechanism aligns well with threat models commonly encountered in IIoT environments.

Attention-Based Robust GNNs. Recent robust GNN defenses use attention mechanisms to reduce adversarial effects by reweighting neighborhood contributions during message passing. PA-GNN combines adversarial training with attention to minimize the impact of unreliable neighbors, whereas GNNGuard uses similarity-based attention and layer-wise memory to block suspicious edges. However, these methods depend on soft attention mechanisms, where all neighbors continue to contribute to the aggregation with different weights. As attention scores are normalized and repeatedly applied across layers, node representations become more similar, leading to excessive feature mixing and over-smoothing [31], specifically under adversarial perturbations. By contrast, AARS uses a DSAM that applies hard Top-K neighbor selection for each node based on edge-level similarity. Specifically, attention scores are calculated through node embedding similarity, and for each node, only the Top-K neighbors with the highest attention scores are retained, whereas all other edges are explicitly masked and do not influence message aggregation. This nodewise hard sparsification enforces a maximum limit on the receptive field at each layer, effectively preventing uncontrolled feature diffusion and reducing over-smoothing. Furthermore, by removing low-similarity edges, the proposed mechanism reduces adversarial connections and minimizes the spread of adversarial gradients during training. A systematic comparison between representative attention-based robust GNN methods is presented in Table 1.

Table 1: Comparison of attention-based robust GNN methods.

Method	Attention Type	Hard Pruning	Node-Wise Top-K	Max Neighbors	Over-Smoothing Control
PA-GNN	Soft	×	×	Unbounded	Weak
GNNGuard	Soft	×	×	Unbounded	Partial
AARS (Ours)	Hard (Top-K)	✓	✓	K	Strong

3 Methodology

Let $G = (V, \mathcal{E}, \mathbf{X})$ be a graph, where V represents a set of N nodes, $\mathcal{E}$ is a set of M edges, and $\mathbf{X} \in \mathbb{R}^{N\times d}$ denotes the node feature matrix, where each row $\mathbf{x}_i$ corresponds to the feature vector of node v_i. The graph structure can be represented using adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, where $\mathbf{A}_{ij} = 1$ if there exists an edge between nodes v_i and v_j, and $\mathbf{A}_{ij} = 0$ otherwise. Degree matrix $\mathbf{D}$ is a diagonal matrix defined as $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$, representing the degree of each node. Given a labeled node set $V_L \subset V$ with corresponding labels Y_L, the objective is to learn a function f_θ that maps nodes to labels and generalizes them well to unlabeled nodes in the graph.

To ensure the consistent propagation of information across the graph, we introduce a normalized propagation matrix $\mathbf{S}$, which is calculated as follows:

$$\mathbf{S} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}} \tag{1}$$

where $\mathbf{I}$ is the identity matrix added to include the self-loops that maintain node-specific information and stabilize feature propagation. This symmetric normalization ensures that the node representations are well-scaled while smoothing the local variations in the feature space.

GNNs follow an iterative message-passing approach, where node representations are gradually updated by collecting information from neighboring nodes. Let $\mathbf{H}^{(0)} = \mathbf{X}$ be an input feature matrix. A generalized layer-wise propagation rule can be expressed as

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)} + \alpha\mathbf{H}^{(l)}\mathbf{W}^{(l)}_{\text{skip}} + \beta\sum_{k=1}^{K}\mathbf{D}^{-1}\mathbf{A}^k\mathbf{H}^{(l)}\mathbf{W}^{(l)}_k\right). \tag{2}$$

Here, $\mathbf{W}^{(l)}$, $\mathbf{W}^{(l)}_{\text{skip}}$, and $\mathbf{W}^{(l)}_k$ denote learnable weight matrices. The residual term weighted by α and the multihop aggregation term weighted by β are introduced to illustrate the commonly used architectural extensions to alleviate oversmoothing and capture long-range dependencies in GNNs.

In this study, we introduced a simplified version of this formulation and do not explicitly include the residual or multihop aggregation terms in the final model. Instead, robustness is improved through stochastic perturbations applied to intermediate node embeddings during training, as formally described in Eq. (6) and examined via ablation studies in Section 4.5. Nonlinear activation function $\sigma(\cdot)$ (e.g., ReLU) guarantees sufficient model expressiveness.

3.1 Adversarial Attacks on GNNs

Adversarial attacks introduce subtle perturbations to data and cause GNNs to perform incorrect classifications, thereby revealing their potential security vulnerabilities. Based on the attack target, adversarial

attacks can be categorized as follows: node-level attacks, which modify node features or local structures; edge-level attacks, which change edge connections; and graph-level attacks, which disrupt the overall structure and features of a graph.

From an attacker's knowledge-level perspective, adversarial attacks can be categorized into different types. White-box attacks assume that an attacker has full knowledge of the model, including its parameters and architecture. Conversely, black-box attacks restrict attackers to accessing only the model's inputs and outputs. Gray-box attacks lie between these two extremes, with the attacker having only partial knowledge of the model [32]. Additionally, adversarial attacks can be classified based on when they occur: evasion attacks occur during testing and attempt to deceive a pretrained model, whereas poisoning attacks occur during training, where malicious changes to the training data affect the learning process of the model [33].

Various attack methods are currently available. Gradient-based attacks, such as FGSM and PGD, manipulate perturbations along a gradient to deceive the model. Optimization-based attacks such as genetic algorithms and particle swarm optimization seek the most effective perturbation strategies. GAN-based attacks use a generator to learn deceptive perturbations that interfere with the model's predictions. Attacks targeting hierarchical graph pooling models combine surrogate and target models to improve attack effectiveness. However, existing attack methods face several challenges. Most methods depend heavily on the gradient information, making them less effective in black-box scenarios. Additionally, specific perturbation strategies are overly obvious, making them easier to detect and defend against. By contrast, our proposed defense method improves the focus of the model on key structures and features, helping it better distinguish between normal and perturbed data, ultimately enhancing its overall robustness.

3.2 Proposed Approach

In adversarial attack scenarios, attackers manipulate edge structures to build attack graphs, causing a significant decline in GNN classification performance [34]. A strong graph that can withstand attacks and maintain an accurate classification is essential for improving the stability of GNNs and boosting node classification reliability.

Given a healthy graph G, node features X, and known partial node label information Y, we aim to develop a graph enhancement approach that creates a more robust graph by boosting the stability of its structure. This process seeks to decrease the vulnerability of the graph in adversarial conditions, thereby enhancing GNN decision-making in hostile environments. This strategy involves dynamically identifying and reinforcing important connections in the graph while repairing potential vulnerable edges. The semantic relationships within the graph are maintained and the known labels Y guide the topological optimization process. This leads to an improved model performance, enabling it to better infer unlabeled nodes. To accomplish this, we propose the AARS, a robust graph learning framework that combines adversarial training and attention mechanisms, as shown in Fig. 1. AARS creates adversarial samples by perturbing both the node features and edge structures. The perturbed samples are combined with the original graph to form multiview representations. A sparse attention mechanism is then used to identify and retain meaningful connections, ultimately reconstructing a more robust graph structure.

AARS combines adversarial training with attention mechanisms to strengthen the graph structure, thereby significantly improving GNN node classification performance in adversarial environments. The next section explains how AARS enhances graph stability and effectively filters out perturbed edges, further exploring its robustness against complex attack scenarios.

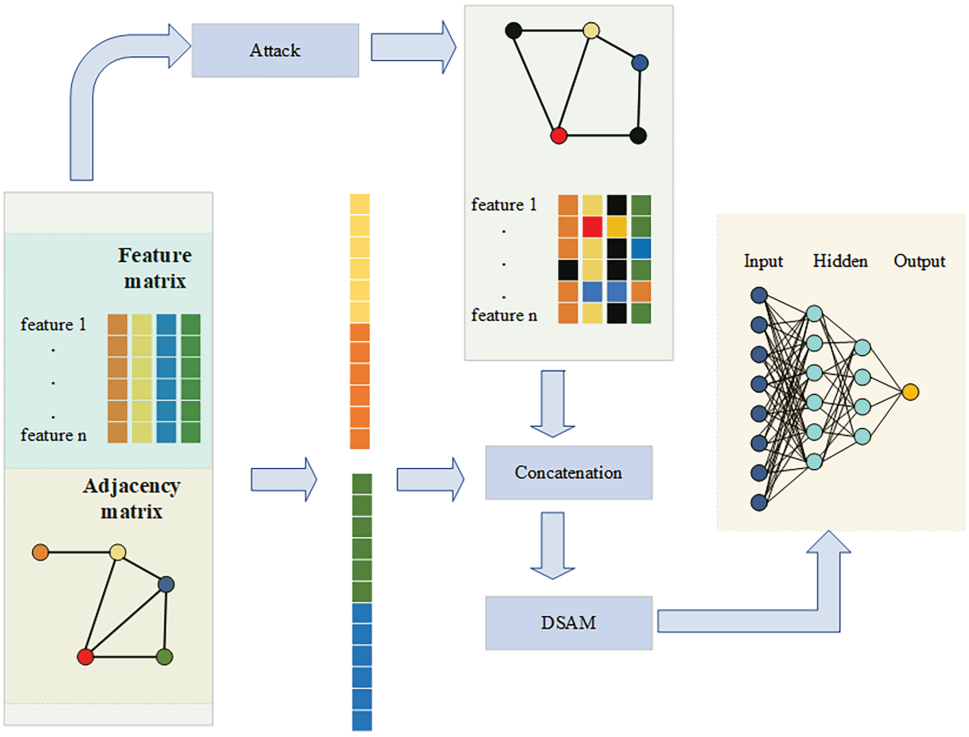


Figure 1: The framework of our proposed AARS is outlined as follows: we first perturb the features and structure of the original graph to generate adversarial examples, which are then concatenated with the original data to form a combined input. This input is fed into a GNN, where an attention-gated mechanism and a dynamic edge selection module are incorporated into the forward pass. Finally, the model jointly optimizes both the standard and adversarial components losses.

3.3 Multiview Adversarial Training Framework

We use Disttack's attack mechanism in reverse for defense, combining multiview adversarial sample creation with gradient synchronization robustness constraints to develop an adaptive, resilient training framework suitable for distributed environments. This approach aims to enhance the adversarial robustness of distributed GNNs, allowing them to effectively withstand disruptions in the structural and feature spaces, thereby improving the stability and generalization of the model in real-world applications.

During each training iteration, we add random perturbations to the 1-hop subgraph $G^{(sub)}$ of the target node to create the corresponding adversarial sample $G'^{(sub)}$ as follows:

$$A'^{(sub)} = A^{(sub)} \oplus \Delta A, \quad X'^{(sub)} = X^{(sub)} \oplus \Delta X \tag{3}$$

Difference from PGD-Based Adversarial Training. It is important to clarify that the proposed randomized adversarial training differs fundamentally from standard PGD-based adversarial training. PGD-based methods explicitly solve an inner maximization problem via gradient-based optimization to generate worst-case adversarial perturbations tailored to the current model. In contrast, the proposed approach does not perform a gradient-based adversarial optimization, nor does it seek worst-case perturbations. Instead, both structural and feature perturbations are generated through randomized, budget-constrained operations that are independent of model gradients. The model is trained to minimize the expected task loss over stochastic perturbations, effectively encouraging robustness under a distribution of perturbations rather than optimizing against a single worst-case direction. This design avoids the computational overhead and

potential training instability associated with iterative adversarial optimization, while still providing diverse perturbation signals that improve robustness against random structural and feature corruptions.

Here, $\oplus$ denotes the randomized perturbations rather than the gradient-based adversarial optimization. ΔA and ΔX represent the perturbations applied to the adjacency matrix (graph structure) and node features, respectively. These perturbations are intended to simulate the effects of attacks on the GNN training process while improving the robustness of the model. Unlike Disttack [23], which directly disrupts the training process, we aim to guide the model through inverse optimization, thereby enabling it to learn adversarially robust feature representations that increase its resistance to similar attacks.

To ensure diversity among adversarial samples, we use a randomized perturbation strategy to generate ΔA and ΔX.

$$\Delta A_{i,j} = \begin{cases} 1 - A^{(sub)}_{i,j}, & \text{with probability } p_{\text{struct}} \text{ (edge flipping)} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

$$\Delta X_{i,j} = \begin{cases} 1 - X^{(sub)}_{i,j}, & \text{with probability } p_{\text{feat}} \text{ (feature flipping)} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$\tilde{X}_{i,j} = X^{(sub)}_{i,j} + \beta \cdot \epsilon_{i,j}, \quad \epsilon_{i,j} \sim \mathcal{N}(0,1), \tag{6}$$

Remark (Perturbation Notation and Scope). In Eq. (4), i, j donates the index nodes in the adjacency matrix $A \in \mathbb{R}^{N \times N}$, whereas in Eq. (5), i denotes the node index and j is the feature dimension in $X \in \mathbb{R}^{N \times d}$. Eq. (5) is used for binary-feature datasets (e.g., Cora and CiteSeer), whereas Eq. (6) applies to continuous-feature datasets (e.g., PubMed). For featureless graphs such as PolBlogs, ΔX is omitted, and DSAM reduces to a structure-only mechanism.

Explanation of Perturbation Design. Eq. (3) provides a conceptual definition of adversarial sample generation, where adversarial subgraphs are created through randomized modifications of the graph structure and node features. Eqs. (4)–(6) specify concrete examples under different data assumptions: Eq. (4) defines structure-level perturbations through probabilistic edge flipping, whereas Eqs. (5) and (6) describe the feature-level perturbations for the binary- and continuous-feature datasets, respectively. These formulations are mutually exclusive and are chosen based on dataset characteristics. In practice, Eqs. (4) and (5) operate on the input adjacency and feature matrices, whereas Eq. (6) injects Gaussian noise into intermediate node embeddings during message passing. All perturbations are randomly generated instead of being gradient-based, functioning as adversarial data augmentation to enhance robustness.

Discussion on Attack Choice and Perturbation Rate. The perturbations defined in Eqs. (4)–(6) are used only during the training phase for adversarial data augmentation rather than as substitutes for strong benchmark attacks.

In contrast to the randomized perturbation strategy adopted in our training framework, gradient-based or optimization-driven attacks such as PGD or Metattack are specifically designed to construct the worst-case perturbations for a given model; however, they are computationally expensive and highly model-dependent. In contrast, our randomized, budget-constrained perturbation strategy enables the efficient generation of diverse adversarial samples without overfitting the model to a specific attack pattern. Where p_{struct} and p_{feat} control the rates of structural and feature perturbations, respectively, and are defined as

$$p_{\text{struct}} = \frac{\text{num\_perturb}}{|E^{(sub)}|}, \quad p_{\text{feat}} = \frac{\text{num\_perturb}}{N^{(sub)} \times F}. \tag{7}$$

Here, $|E^{(sub)}|$, $N^{(sub)}$, and F denote the number of edges in the subgraph, number of nodes, and feature dimension, respectively. Parameter num_perturb specifies the expected number of perturbations used to compute the per-edge and per-feature perturbation probabilities.

To avoid excessive modifications that could significantly alter the original graph, we control the perturbation scale by applying rate-based budget constraints separately to the graph structure and node features. Specifically, the numbers of perturbed edges and perturbed feature entries are bounded as

$$\|\Delta A\|_0 \le P_{\text{struct}}, \quad \|\Delta X\|_0 \le P_{\text{feat}}, \tag{8}$$

where

$$P_{\text{struct}} = p \cdot |E^{(sub)}|, \quad P_{\text{feat}} = p \cdot \left(N^{(sub)} \times F\right) \tag{9}$$

Here, p is a global perturbation rate controlling the overall perturbation scale relative to the size of the sampled subgraph. In this framework, p_{struct} and p_{feat} define the expected sampling rates for randomized structural and feature perturbations, respectively, and P_{struct} and P_{feat} act as strict upper bounds. The perturbation rate, p, is treated as a fixed hyperparameter in our experiments and is set to a small value following the standard practice in adversarial graph training.

To examine the effect of p on model robustness, we conduct experiments by varying p across a range of values. As shown in subsequent experiments, AARS maintains a stable and high performance within a reasonable range of p, demonstrating that the rate-based budget effectively constrains perturbations without overfitting to a specific attack pattern. Perturbations are randomly sampled according to p_{struct} and p_{feat} under these budget constraints, providing diverse adversarial signals that improve the robustness while preserving the original graph structure.

This method implicitly preserves the original data distribution, enabling adversarial samples to enhance the robustness of the model without significantly deviating from the original dataset, thereby ensuring training stability. To fully utilize the adversarial samples, we combine the original data G with the adversarial samples G' to create a multiview input $G_{\text{multi}=\{G,G'\}}$, as shown in Fig. 2.

This figure shows how multiview input $G_{\text{multi}} = \{G, G'\}$ is built, where $G = (X, A)$ represents the original graph and $G' = (X', A')$ is the adversarially modified version. The perturbation process causes feature and structural changes that mimic potential attacks, thereby creating different graph views.

To create the multiview input, node feature matrices X and X' are combined along the feature dimension, resulting in $X_{\text{multi}} = [X \,|\, X']$. Meanwhile, adjacency matrices A and A' remain separate to maintain the structural integrity of each graph. During training, the model processes (X, A) and (X', A'), allowing for multiview feature aggregation and improved robustness while preserving the topology of each graph.

During training, the model processes both the original graph data and perturbed adversarial samples simultaneously, learning more resilient feature representations through gradient updates.

To balance task-specific optimization with improved adversarial robustness, we design a hybrid loss function defined as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}}(f_\theta(G), Y) + \lambda \cdot \mathcal{L}_{\text{task}}(f_\theta(G'), Y) \tag{10}$$

Here, λ is a weighting hyperparameter that balances the contributions of the clean and adversarially perturbed samples during training. A larger λ promotes robustness by emphasizing adversarial examples, whereas a smaller λ focuses more on clean data performance. In our implementation, λ is regarded as a tunable hyperparameter and is selected based on the validation results.

where:

- $\mathcal{L}_{\text{task}}$ denotes the task-specific loss (e.g., cross-entropy for node classification).
- G and G' represent the original graph and its adversarially perturbed counterpart, respectively.

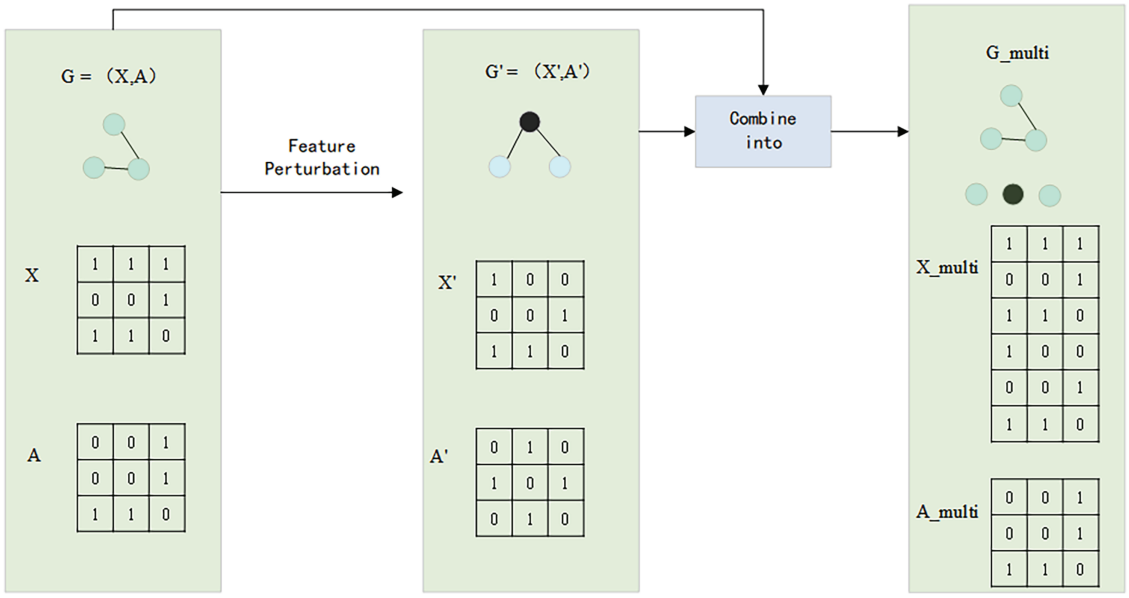


Figure 2: Multiview adversarial sample generation process.

3.4 Dynamic Sparse Attention Mechanism

We propose a dynamic sparsification attention mechanism that reduces computational redundancy and suppresses adversarial noise propagation using a Top-K [35] selection strategy. In this mechanism, we dynamically choose the Top-K most important edges based on node feature similarity, retaining only the most representative neighbor connections to lower the impact of low-weight, noisy edges [36]. Specifically, we begin by calculating the attention score for each edge, which is defined as the dot product of the node features.

$$\alpha_{i,j} = \frac{x_i \cdot x_j}{\sqrt{d}} \tag{11}$$

The overall process of computing the attention scores and performing Top-K selection is illustrated in Fig. 3.

This figure shows how DSAM works. Starting with a perturbed graph that includes both real (blue solid lines) and adversarial (red dashed lines) edges, the model first calculates the node similarity to produce attention scores for each edge. Edge thickness indicates the strength of the score, with thicker lines indicating higher attention scores. Using these scores, a Top-K selection process maintains the most representative and semantically meaningful neighbors while removing low scores or adversarial edges. The resulting optimized sparse graph maintains important structural information and effectively reduces the spread of noisy perturbations, thereby enhancing message passing robustness.

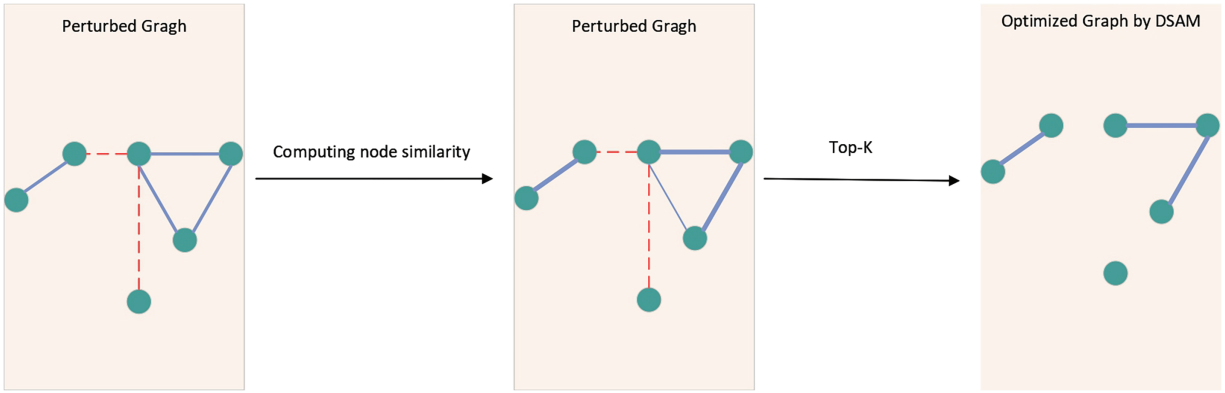


Figure 3: Dynamic sparse attention mechanism.

Where d denotes the feature dimension. To enable dynamic sparsification, we retained only the Top-K most similar edges for each node v_i, filtering out the remainder, which effectively eliminates adversarial disturbances during message passing.

$$\mathcal{N}^{(k)}(v_i) = \{v_j \in \mathcal{N}(v_i) \mid \alpha_{i,j} \text{ is among the top } K \text{ highest values}\} \tag{12}$$

Thus, the message aggregation process can be simplified as follows:

$$h_i^{(l+1)} = \sigma\left(\sum_{v_j \in \mathcal{N}^{(k)}(v_i)} \alpha_{i,j} W^{(l)} h_j^{(l)}\right) \tag{13}$$

This mechanism reduces the computational complexity of attention from $O(N^2)$ to $O(N \cdot K)$ $(K \ll N)$, improving efficiency while enhancing adversarial robustness. The Top-K selection preserves high-similarity connections, suppressing adversarial edges, and reduces the propagation of the adversarial gradient $\nabla_A \mathcal{L}_{\text{atk}}$, thereby making parameter updates less sensitive to adversarial perturbations. Algorithm 1 shows that this dynamic sparsification strategy is integrated into the training loop. Additionally, this method improves distributed training by requiring each computing node to synchronize only the information of the Top-K neighbors, thereby reducing communication overhead and avoiding load imbalance.

Furthermore, this mechanism complements multiview adversarial training; during training, it enhances generalization against perturbations, and during inference, it filters out adversarial edges, further boosting model robustness. The dynamic sparsification can be formally described as a constraint.

$$|\mathcal{N}^{(k)}(v_i)| \leq K, \quad \forall v_i \tag{14}$$

Here, K is a tunable hyperparameter for sparsification that controls the size of the local neighborhood during message passing. Smaller values of K enforce stronger sparsification, which suppresses potentially adversarial edges, whereas larger values maintain more structural information but increase sensitivity to perturbations. To validate the effectiveness of Top-K sparsification, we conduct additional experiments on GPU memory usage and runtime efficiency, which are reported in Section 4.6, the peak GPU memory usage and average per-epoch runtime of AARS and representative robust GNN baselines on Cora and PubMed. AARS consistently uses less memory and runs faster, demonstrating practical efficiency improvements beyond mere asymptotic complexity reduction. Additionally, the method remains stable across a reasonable

range of K and its performance is not significantly affected by the specific choice of this parameter (see Section 4.5 for a detailed analysis).

Algorithm 1: Adversarial robust graph neural network training

Require: Node features $\mathbf{X}$, adjacency matrix $\mathbf{A}$, labels $\mathbf{y}$, perturbation rate p, attention threshold K
Ensure: Trained model parameters Θ
1: Initialize Θ, create adversarial attack generator $\mathcal{D}$
2: **for** each iteration $t = 1$ to T **do**
3: Generate adversarial samples: $\mathbf{X}_{adv}, \mathbf{A}_{adv} \leftarrow \mathcal{D}(\mathbf{X}, \mathbf{A}, \varepsilon)$
4: Combine data: $\mathbf{X}_{comb} \leftarrow [\mathbf{X}; \mathbf{X}_{adv}], \mathbf{A}_{comb} \leftarrow [\mathbf{A}; \mathbf{A}_{adv}]$
5: Graph convolution encoding: $\mathbf{H} \leftarrow \mathrm{GCN}(\mathbf{X}_{comb}, \mathbf{A}_{comb})$
6: Build sparse attention graph: select Top-K neighbors $\mathbf{A}_{attn}$
7: Adversarially robust prediction: $\hat{\mathbf{y}} \leftarrow \mathrm{GCN}(\mathbf{H}, \mathbf{A}_{attn})$
8: Compute loss: $\mathcal{L}_{total} \leftarrow \mathcal{L}_{cls} + \lambda \mathcal{L}_{adv}$
9: Update parameters: $\Theta \leftarrow \Theta - \eta \nabla_{\Theta} \mathcal{L}_{total}$
10: **end for**
11: **return** Θ

In Algorithm 1, adversarial samples (X_{adv}, A_{adv}) generated in line 3 correspond to perturbed graph (X', A') in our formulation, produced by the adversarial attack generator. In line 4, combined graph (X_{comb}, A_{comb}) represents the multigraph (X_{multi}, A_{multi}), which is created by concatenating the original clean and adversarial graphs. Line 6 introduces A_{attn}, which is the attention-based sparse adjacency matrix obtained by retaining only the Top-K most informative neighbors for each node to improve robustness. Finally, loss term $\mathcal{L}_{cls} + \lambda \cdot \mathcal{L}_{adv}$ in line 8 refers to the hybrid loss defined in Eq. (9), where λ balances the contributions of clean and adversarial samples during training.

4 Experiments

4.1 Dataset Setup

Our experiments were conducted on six popular graph datasets, including the citation networks Cora, CiteSeer, PubMed, the blog and citation network PolBlogs, as well as the extended datasets Cora_ML and the large-scale OGB dataset OGBN-Arxiv, as summarized in Table 2. Notably, the PolBlogs dataset does not include node feature information. For all experiments, we used the Adam optimizer with an initial learning rate of 0.01. Each model was trained for 200 epochs with a hidden layer size of 16. To minimize the effects of random initialization, each experiment was repeated ten times. The nodes were divided as follows: 10% for training, 10% for validation, and 80% for testing. The inclusion of Cora_ML and OGBN-Arxiv enables us to evaluate our method on both mid-sized and large-scale graphs, offering a more comprehensive assessment of the performance and robustness across different graph sizes and feature types.

4.2 Implementation Details of AARS

We use the Adam optimizer (learning rate 0.01, weight decay 5e–4) to jointly optimize all model parameters, with adaptive Gaussian noise added after the first GCN layer to improve the robustness against perturbations. During training, the adversarial adjacency matrices produced by the attack methods are combined with the original data for joint optimization. Additionally, structural pruning based on feature similarity is performed to maintain important neighbor connections. The model is trained for 200 epochs with dropout (rate of 0.5), L2 regularization (5e–4), and ReLU activation after the first layer. Experiments

are run on a system with an NVIDIA A100 Tensor Core GPU, an AMD EPYC 7742 CPU (2.25 GHz), and 1 TB of RAM. For robustness optimization in semi-supervised adversarial settings, we calculate node feature similarity using the Jaccard index on the Cora, CiteSeer, and PolBlogs datasets, and cosine similarity on the PubMed dataset. Different similarity metrics are used for various datasets to match node features, ensuring that the structural optimization step can accurately evaluate node similarity and enhance the model's robustness under adversarial attacks. Edge weight selection is improved through feature inner product calculations to enable dynamic structural optimization.

Table 2: Dataset statistics.

	Nodes	Edges	Classes	Features	Feature Type
Cora	2485	5069	7	1433	Binary
CiteSeer	2110	3668	6	3703	Binary
PolBlogs	1222	16,714	2	–	–
PubMed	19,717	44,338	3	500	Continuous
Cora_ML	2710	5278	7	1433	Binary
OGBN-Arxiv	169,343	1,166,243	40	128	Continuous

4.3 Comparative Analysis with State-of-the-Art Methods

We evaluate multiple GNN defense methods using the GCN model under different attack budgets (ϵ), including structural (Metattack and DICE) and feature-level attacks. For feature attacks, given node feature matrix $X \in \mathbb{R}^{N\times F}$, we randomly flip $\epsilon \cdot N \cdot F$ feature entries ($0 \leftrightarrow 1$), simulating feature noise without gradient-based optimization. Here, ϵ represents the fraction of edges perturbed for Metattack/DICE and the fraction of node features flipped for the Feature Attack. The detailed results are presented in Table 3.

Table 3: Classification accuracy (mean ± standard deviation) of Guard [20], Jaccard [12], SVD [13], Noisy [37], RGNN [38], and AARS on benchmark node classification datasets under attack budget ϵ. Here, ϵ denotes the fraction of edges perturbed for Metattack/DICE and the fraction of node features flipped for Feature Attack. The best result in each row is highlighted in bold.

Dataset	ε	Guard	Jaccard	SVD	RGNN	Noisy	AARS
				Metattack			
Cora	0%	77.5 ± 0.7	80.9 ± 0.7	80.6 ± 0.4	83.5 ± 0.3	83.2 ± 0.4	**86.0 ± 0.6**
	5%	75.8 ± 0.6	78.9 ± 0.8	78.4 ± 0.6	78.3 ± 0.6	81.2 ± 0.7	**83.7 ± 0.5**
	10%	74.7 ± 0.4	76.7 ± 0.7	71.5 ± 0.8	70.7 ± 0.8	74.5 ± 0.6	**80.4 ± 0.4**
CiteSeer	0%	70.1 ± 1.5	71.2 ± 0.7	70.7 ± 0.4	72.3 ± 0.5	71.9 ± 0.4	**75.4 ± 0.6**
	5%	69.9 ± 1.1	70.3 ± 2.3	68.9 ± 0.7	70.6 ± 0.7	72.3 ± 0.6	**73.0 ± 0.4**
	10%	70.0 ± 1.5	67.5 ± 2.1	68.8 ± 0.6	68.7 ± 1.2	70.4 ± 0.8	**70.6 ± 0.5**
PubMed	0%	84.5 ± 0.6	85.0 ± 0.5	82.7 ± 0.3	85.1 ± 0.8	85.0 ± 0.6	**86.1 ± 0.5**
	5%	**84.3 ± 0.9**	79.6 ± 0.3	81.3 ± 0.6	81.1 ± 0.7	81.8 ± 0.4	83.3 ± 0.7
	10%	**84.1 ± 0.3**	67.4 ± 1.1	81.1 ± 0.7	65.2 ± 0.4	73.3 ± 0.6	82.3 ± 0.6
PolBlogs	0%	93.1 ± 0.6	–	86.5 ± 0.8	94.9 ± 0.3	**95.2 ± 0.4**	93.4 ± 0.4
	5%	72.8 ± 0.8	–	**85.1 ± 1.6**	76.0 ± 0.8	79.7 ± 0.6	83.5 ± 0.5
	10%	68.7 ± 1.0	–	**84.8 ± 2.3**	69.2 ± 1.2	73.4 ± 0.5	80.9 ± 0.6

(Continued)

Table 3 (continued)

Dataset	ε	Guard	Jaccard	SVD	RGNN	Noisy	AARS
Cora-ML	0%	85.9 ± 0.6	84.9 ± 0.5	82.4 ± 0.7	86.3 ± 0.4	85.2 ± 0.5	**86.7 ± 0.3**
	5%	81.0 ± 0.4	80.6 ± 0.7	82.1 ± 0.5	81.4 ± 0.6	79.4 ± 0.8	**83.3 ± 0.4**
	10%	75.0 ± 0.7	75.4 ± 0.5	**80.7 ± 0.6**	75.2 ± 0.4	73.1 ± 0.7	75.9 ± 0.5
DICE							
Cora	5%	76.4 ± 0.4	79.6 ± 0.6	74.9 ± 1.3	76.9 ± 0.6	82.5 ± 0.8	**85.0 ± 0.6**
	10%	76.6 ± 0.5	78.6 ± 0.8	73.5 ± 1.5	80.0 ± 0.6	80.5 ± 0.6	**83.5 ± 0.6**
CiteSeer	5%	68.5 ± 1.6	70.9 ± 0.4	69.4 ± 1.6	69.3 ± 0.5	70.8 ± 0.3	**74.1 ± 0.3**
	10%	69.9 ± 1.5	69.9 ± 0.6	68.1 ± 1.5	67.8 ± 1.1	70.4 ± 0.8	**72.8 ± 0.6**
PubMed	5%	84.0 ± 0.8	83.4 ± 0.7	81.5 ± 0.8	83.8 ± 0.6	83.6 ± 0.9	**85.5 ± 1.1**
	10%	83.6 ± 1.0	81.8 ± 0.5	81.4 ± 0.5	82.4 ± 0.8	82.1 ± 2.3	**84.0 ± 2.1**
PolBlogs	5%	81.3 ± 0.7	–	86.5 ± 2.3	89.6 ± 0.4	**90.3 ± 0.3**	89.8 ± 0.6
	10%	78.9 ± 0.6	–	85.3 ± 2.8	85.5 ± 0.9	86.1 ± 0.9	**86.5 ± 0.7**
Feature Attack							
Cora	0%	83.2 ± 0.6	82.0 ± 0.5	78.9 ± 0.7	83.2 ± 0.6	82.0 ± 0.5	**85.3 ± 0.6**
	5%	80.0 ± 0.7	79.7 ± 0.5	70.2 ± 0.9	80.0 ± 0.6	76.4 ± 0.8	**82.3 ± 0.5**
	10%	74.5 ± 0.8	74.5 ± 0.7	68.5 ± 0.6	74.6 ± 0.9	70.8 ± 0.5	**79.7 ± 0.6**
CiteSeer	0%	72.2 ± 0.6	72.5 ± 0.5	69.5 ± 0.7	72.2 ± 0.6	71.9 ± 0.6	**74.7 ± 0.6**
	5%	65.7 ± 0.6	66.1 ± 0.8	59.4 ± 0.7	65.7 ± 0.5	65.7 ± 0.6	**70.6 ± 0.7**
	10%	62.3 ± 0.7	62.3 ± 0.9	53.9 ± 0.5	62.4 ± 0.6	59.7 ± 0.8	**66.5 ± 0.6**
PubMed	0%	85.9 ± 0.7	85.9 ± 0.6	84.4 ± 0.7	85.1 ± 0.6	84.9 ± 0.7	**86.2 ± 0.6**
	5%	60.4 ± 0.5	60.1 ± 0.6	53.5 ± 0.8	52.4 ± 0.7	54.3 ± 0.9	**61.7 ± 0.8**
	10%	58.0 ± 0.9	**58.0 ± 0.7**	52.8 ± 0.6	48.8 ± 0.8	52.7 ± 0.7	55.1 ± 0.6
Cora-ML	0%	85.5 ± 0.7	84.7 ± 0.6	82.3 ± 0.7	85.5 ± 0.6	85.3 ± 0.6	**86.5 ± 0.6**
	5%	74.7 ± 0.6	74.0 ± 0.7	68.9 ± 0.8	74.5 ± 0.5	69.7 ± 0.9	**75.4 ± 0.7**
	10%	**73.7 ± 0.8**	73.4 ± 0.6	70.5 ± 0.9	73.4 ± 0.7	67.7 ± 0.8	70.5 ± 0.6
OGBN-Arxiv	0%	58.7 ± 0.6	58.3 ± 0.5	–	–	55.1 ± 0.7	**59.5 ± 0.6**
	5%	50.1 ± 0.6	49.1 ± 0.5	–	–	47.3 ± 0.7	**52.4 ± 0.9**
	10%	47.1 ± 0.7	41.0 ± 0.6	–	–	43.2 ± 0.5	**49.1 ± 0.8**
Feature Attack + Metattack							
Cora	0%	82.7 ± 0.6	81.8 ± 0.7	78.1 ± 0.5	83.1 ± 0.6	81.8 ± 0.5	**85.3 ± 0.7**
	5%	73.9 ± 0.7	73.6 ± 0.5	71.3 ± 0.8	70.4 ± 0.6	69.7 ± 0.7	**75.2 ± 0.6**
	10%	64.3 ± 0.8	64.3 ± 0.6	67.7 ± 0.7	59.5 ± 0.5	61.2 ± 0.6	**72.1 ± 0.7**
CiteSeer	0%	73.2 ± 0.6	73.0 ± 0.7	70.0 ± 0.5	70.5 ± 0.6	72.1 ± 0.7	**75.4 ± 0.6**
	5%	64.9 ± 0.7	65.1 ± 0.8	59.7 ± 0.6	61.7 ± 0.7	62.7 ± 0.6	**66.2 ± 0.7**
	10%	25.1 ± 0.5	58.5 ± 0.6	53.5 ± 0.7	48.8 ± 0.6	52.2 ± 0.7	**59.7 ± 0.6**

(Continued)

Table 3 (continued)

Dataset	ε	Guard	Jaccard	SVD	RGNN	Noisy	AARS
Cora-ML	0%	85.5 ± 0.7	85.0 ± 0.6	81.9 ± 0.7	86.3 ± 0.6	83.9 ± 0.5	**87.2 ± 0.6**
	5%	65.0 ± 0.6	64.9 ± 0.7	69.0 ± 0.8	65.6 ± 0.5	62.7 ± 0.7	**70.5 ± 0.6**
	10%	56.4 ± 0.7	56.4 ± 0.6	**68.3 ± 0.8**	60.4 ± 0.6	50.3 ± 0.7	62.8 ± 0.6

AARS consistently outperforms or is highly competitive across all datasets and perturbation levels. On Cora, it achieves 86.0% accuracy without perturbation and maintains strong performance under 5% and 10% perturbations (83.7% and 80.4%, respectively), surpassing Guard (75.8%, 74.7%) and RGNN (78.3%, 70.7%). On CiteSeer, AARS leads under 0% and 5% perturbations (75.4%, 73.0%) and remains competitive at 10% (70.6% vs. 70.0% by Guard). For PubMed, although Guard slightly outperforms AARS at 5% and 10% perturbations, AARS still surpasses Jaccard and RGNN, demonstrating good generalization. On PolBlogs, AARS achieves 83.5% at 5% perturbation, outperforming RGNN by more than seven points, and remains competitive at 10%. On Cora-ML, AARS achieves the best performance under 0% and 5% perturbations (86.7%, 83.3%), slightly trailing SVD at 10% (75.9% vs. 80.7%), indicating robust adaptability to varying graph structures.

Under DICE attacks, AARS also demonstrates superior performance. On Cora, it achieves 85.0% and 83.5% under 5% and 10% perturbations, respectively, outperforming Guard (76.4%, 76.6%) and SVD (74.9%, 73.5%). On CiteSeer, AARS maintains leading accuracy (74.1%, 72.8%), whereas on PubMed, it achieves the highest results (85.5%, 84.0%), even under feature-level perturbations. On PolBlogs, AARS reaches 86.5% at 10% perturbation, slightly surpassing noisy (86.1%) and RGNN (85.5%).

AARS exhibits significant robustness against feature perturbations. On Cora, it achieves 85.3% at 0% and maintains 82.3% and 79.7% under 5% and 10% feature flips, respectively, consistently outperforming other baselines. On CiteSeer, AARS achieves the best results at all perturbation levels (74.7%, 70.6%, 66.5%), whereas on PubMed, it reaches 86.2% at 0% and 61.7% at 5% perturbation, outperforming all other defenses. On Cora-ML, AARS maintains leading performance at 0% and 5% perturbations, and slightly trails SVD at 10% (70.5% vs. 73.7%). On the large-scale OGBN-Arxiv dataset, AARS achieves the highest accuracy under all perturbation levels (59.5%, 52.4%, and 49.1%), demonstrating scalability and robustness on large graphs.

When feature attacks are combined with Metattack, AARS still shows strong resilience. On Cora, it achieves 85.3% at 0% and maintains 75.2% and 72.1% under 5% and 10% combined perturbations, respectively, outperforming all baselines. On CiteSeer and Cora-ML, AARS similarly leads under most perturbation levels, showing its effectiveness in defending against both structural- and feature-level attacks simultaneously.

Overall, AARS demonstrates strong robustness against Metattack, DICE, Feature Attack, and their combinations, highlighting its generalization capability and practical value. In contrast, existing baselines have limitations: Guard handles local similarity but struggles under global perturbations; Jaccard and SVD rely on fixed edge pruning or low-rank approximations and may discard true edges; RGNN focuses only on local smoothness; and Noisy provides nontargeted noise injection with limited protection against structured or distributed attacks. A portion of the comparison results was obtained from [37], and the experiments are reproduced using the original parameter settings to ensure fairness and credibility.

To illustrate robustness under increasing attack budgets, Fig. 4 shows F1 and AUC [39,40] on Cora and CiteSeer. Baselines degrade noticeably, specifically under combined attacks, whereas AARS consistently maintains a high performance, complementing Table 3.

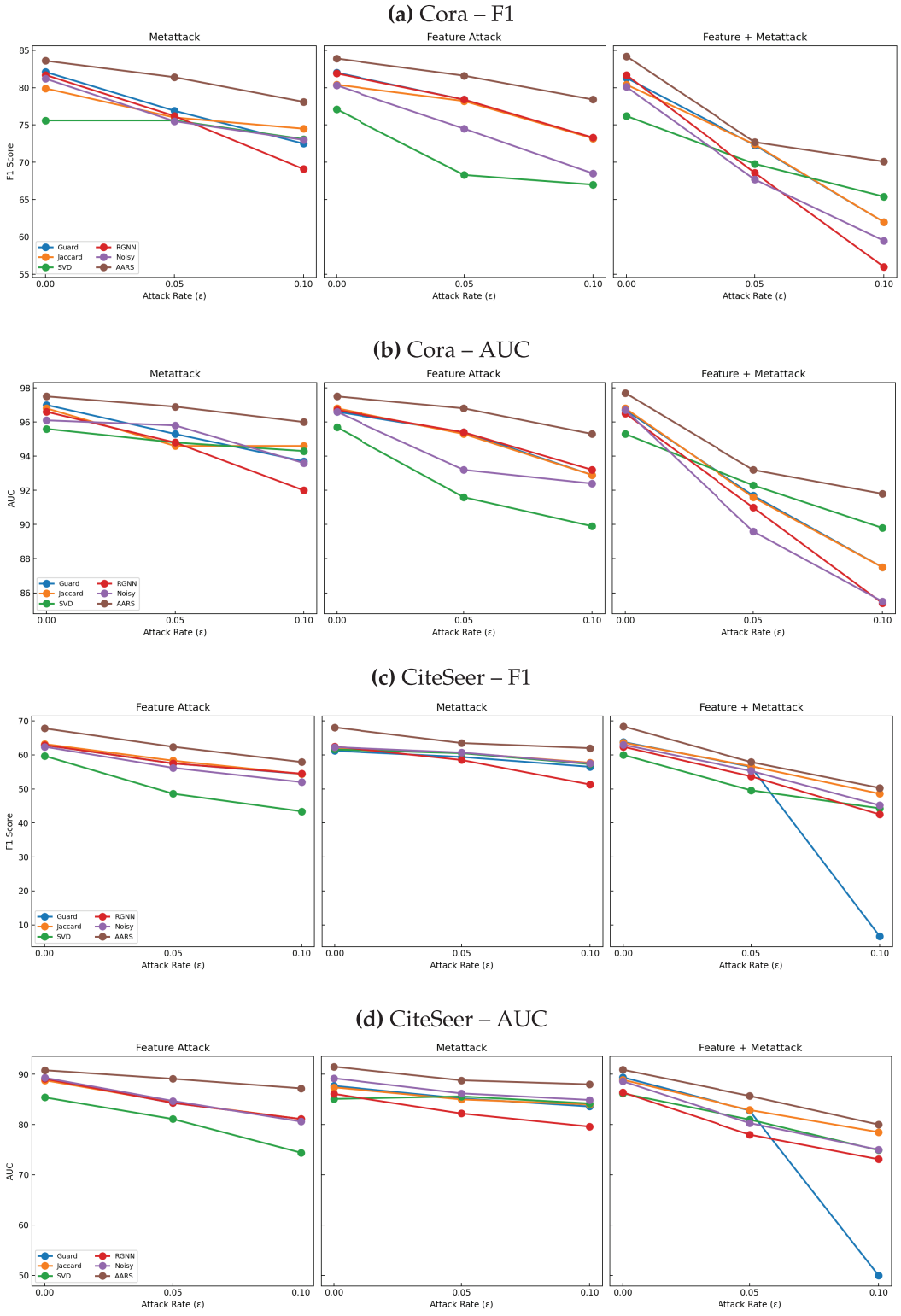


Figure 4: Performance trends of different defense methods under various structural attack scenarios. X-axis: attack budget ϵ; Y-axis: performance metric (F1 or AUC). Three attack types are plotted: Metattack, Feature Attack, and Feature Attack + Metattack.

To further validate the effectiveness of AARS, we compare it with recent state-of-the-art GNN defense methods, including Mid-GCN [41], L2M-GCN [42], and RCNLip [43], on Cora, CiteSeer, and PubMed. As shown in Table 4, AARS consistently achieves the best or highly competitive performance across different attack budgets, demonstrating its superiority even against the latest adversarial defense techniques.

Table 4: Comparison of AARS with recent state-of-the-art methods on Cora, CiteSeer, and PubMed under Metattack. The best result in each row is highlighted in bold.

Dataset	ε	Mid-GCN	L2M-GCN	RCNLip	AARS
			Metattack		
Cora	0%	83.1	82.7	83.6	**86.0**
	5%	82.6	79.7	78.9	**83.7**
	10%	79.1	79.3	72.8	**80.4**
CiteSeer	0%	73.5	73.2	72.8	**75.4**
	5%	72.6	72.1	72.0	**73.0**
	10%	71.0	72.2	68.9	**70.6**
PubMed	0%	85.6	85.9	85.7	**86.1**
	5%	83.5	84.8	81.0	**83.3**
	10%	81.7	84.8	77.7	**82.3**

Note: Values for Mid-GCN, L2M-GCN, and RCNLip are cited from [42,44].

4.4 Analyses for Adversarial Training with Varying Perturbation Rates

To examine how the strength of adversarial perturbations affects model performance, we perform a series of experiments by adjusting the perturbation rate during adversarial training. Fig. 5 shows the test accuracy of the AARS model at different perturbation rates from 0.0 to 1.0. Importantly, the highest test accuracy is achieved at a perturbation rate of 0.1, surpassing 80.4%. This finding underscores that selecting the correct perturbation magnitude is essential for balancing robustness and generalization.

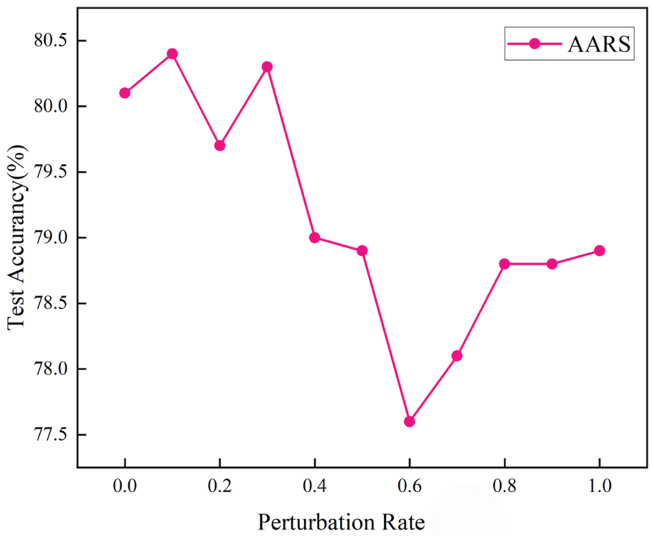


Figure 5: Test accuracy of AARS under different perturbation rates.

When the perturbation rate is too low (e.g., 0.0 or 0.05), the adversarial examples created during training are not sufficiently challenging. Consequently, the model tends to overfit clean training data and does not learn representations resistant to even mild adversarial noise. Conversely, when the perturbation rate becomes too high (e.g., 0.6 or above), the excessive perturbations significantly distort the original graph structure or node features, impairing the ability of the model to maintain semantic relationships and generalize to clean data. This causes a noticeable decline in the test accuracy, specifically at the 0.6 perturbation level.

The better performance at the 0.1 setting indicates that a moderate perturbation rate offers an optimal adversarial signal. This creates meaningful changes that help the model develop more robust representations while retaining the core features of the input graph. This highlights the importance of carefully adjusting the perturbation rates in adversarial training because both extremely little and much perturbation can harm the model's performance.

4.5 Ablation Studies

To assess the effectiveness of each key module in AARS, we performed ablation experiments on the Cora dataset under Metattack (a targeted attack), comparing the robustness of the model across the different module configurations. The results are summarized in Table 5.

Table 5: Comparison of baseline methods and AARS under Metattack on Cora, CiteSeer, and PolBlogs. The best result in each row is highlighted in bold.

Dataset	ε	Guard	Jaccard	SVD	RGNN	Noisy	AARS-A	AARS-T	AARS-ALL
					Metattack				
Cora	0%	77.5	80.9	80.6	83.5	83.2	85.8	82.9	**85.9**
	5%	75.8	78.9	78.4	78.3	81.2	82.8	79.9	**83.7**
	10%	74.7	76.7	71.5	70.7	74.5	80.3	76.0	**80.4**
CiteSeer	0%	70.1	71.2	70.7	72.3	71.9	75.4	75.3	**75.4**
	5%	69.9	70.3	68.9	70.6	72.3	72.3	**73.4**	73
	10%	70.0	67.5	68.8	68.7	70.4	**70.8**	70.2	70.6
PolBlogs	0%	93.1		86.5	94.9	**95.2**	93.9	93.5	93.4
	5%	72.8		**85.1**	76.0	79.7	81.4	83.3	83.5
	10%	68.7		**84.8**	69.2	73.4	75.1	80.0	80.9

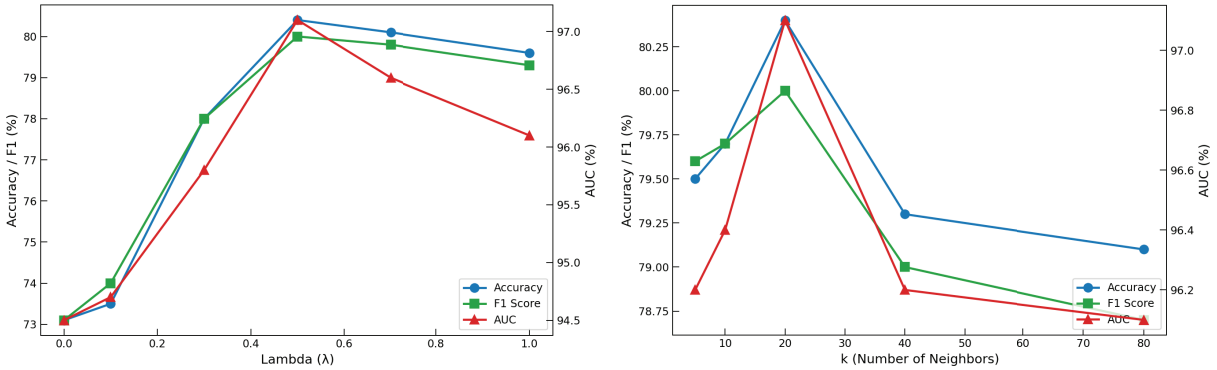
Specifically, AARS-A refers to a model that uses adversarial training alone without the attention mechanism; AARS-T includes the attention mechanism but excludes adversarial training; and AARS-ALL represents the full model, combining both adversarial training and the attention mechanism. As shown in Table 5, the AARS-ALL model consistently achieves the best performance across all attack budgets, with accuracies of 85.9%, 83.7%, and 80.4%, significantly outperforming both the ablated versions and other baseline methods.

Compared to the others, AARS-T, which omits adversarial training, shows significant performance swings, with accuracy declining to 76.0% under a 10% perturbation level, highlighting the crucial role of adversarial training in maintaining model stability. However, AARS-A, which only removes the attention mechanism, experiences a decrease in accuracy to 80.3% at the same 10% attack level, indicating that the attention mechanism is also vital for improving model robustness.

Therefore, the superior performance of AARS is owing to the combined boost from both the attention mechanism and adversarial training—neither can be left out. The full model remains strong even during severe attacks, demonstrating its high adaptability and usefulness in real-world graph-disruption cases.

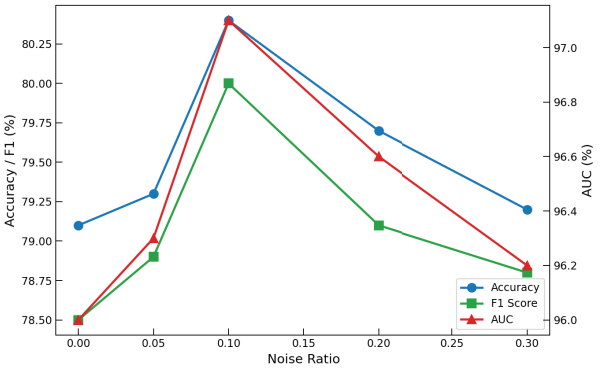
Building on the analysis of perturbation rates, we examined how the key hyperparameters of AARS—λ, k, and noise ratio—affect its robustness under a targeted Metattack with $\epsilon = 0.1$ on the Cora dataset.

Varying λ from 0 to 1 (Fig. 6a) shows that the Accuracy, F1, and AUC increase up to $\lambda = 0.5$, reaching 80.4%, 80.0%, and 97.1%, respectively, followed by a slight decline, indicating that an intermediate weighting of adversarial training and attention components is optimal.



(a) Effect of λ

(b) Effect of k



(c) Effect of noise ratio

Figure 6: Hyperparameter and noise sensitivity analysis of AARS on Cora.

Increasing the number of neighbors k from 5 to 20 (Fig. 6b) improves all metrics, with peaks at accuracy 80.4%, F1 80.0%, and AUC 97.1%. Further increases cause minor declines, suggesting that the top 20 neighbors provide sufficient context while avoiding noise.

Finally, testing different feature noise ratios from 0 to 0.3 (Fig. 6c) shows stable performance under low noise (0–0.1) and gradual decreases beyond 0.1, with peak values similar to λ and k.

Overall, these results indicate that AARS maintains high robustness and controls the sensitivity to hyperparameters under adversarial attacks.

4.6 Efficiency and Runtime Analysis

To evaluate the practical efficiency of AARS, we measured both the peak GPU memory usage and inference runtime on the Cora and PubMed datasets. The results are summarized in Table 6, compared against several baseline methods, including NoisyGCN, RGCN, GCN-Jaccard, GCN-SVD, and GNNGuard.

Table 6: Peak GPU Memory and inference runtime of different GNN methods.

Method	Peak GPU Memory (PubMed)	Peak GPU Memory (Cora)	Runtime (PubMed)	Runtime (Cora)
AARS (Top-K)	1.55 GB	43.42 MB	3.8 ms	0.4 ms
NoisyGCN	1.60 GB	59.45 MB	11.9 ms	5.6 ms
RGCN	10.47 GB	200.15 MB	332.4 ms	10.5 ms
GCN-Jaccard	1.61 GB	46.67 MB	22.1 ms	9.4 ms
GCN-SVD	7.48 GB	138.95 MB	366.3 ms	22.2 ms
GNNGuard	1.60 GB	59.45 MB	5.9 ms	2.7 ms

As shown in the table, AARS (Top-K) significantly reduces memory consumption compared to methods with full adjacency computations such as RGCN and GCN-SVD. For example, on PubMed, AARS requires 1.55 GB of GPU memory, whereas RGCN consumes 10.47 GB and GCN-SVD 7.48 GB. Similar trends are observed on Cora, demonstrating the practical effect of reducing computational complexity from $O(N^2)$ to $O(N \cdot K)$.

In terms of inference runtime, AARS is also highly efficient. On PubMed, AARS takes only 3.8 ms per forward pass, compared to 332.4 ms for RGCN and 366.3 ms for GCN-SVD. On Cora, the runtime is reduced to 0.4 ms, showing that our Top-K strategy achieves substantial speedups without sacrificing performance.

Overall, these results confirm that AARS not only maintains strong robustness under adversarial attacks, but also offers considerable practical efficiency in terms of both memory usage and inference speed, effectively addressing the theoretical complexity reduction in real-world scenarios.

5 Limitations and Discussions

Although AARS shows notable defensive benefits against attacks such as Metattack and DICE on datasets such as Cora and CiteSeer, it still faces important challenges related to robustness and scalability. AARS uses a sparse attention mechanism based on feature similarity (see Algorithm 1) to dynamically filter neighbors and reduce adversarial edge propagation. However, in cases where the graph structure is heavily degraded or node features are significantly altered, noise can easily distort the feature space, leading to biased attention weights. This causes adversarial connections to be mistakenly maintained or strengthened. While the sparse attention mechanism decreases the per-layer computational complexity from $O(N^2)$ to $O(NK)$, where K is the Top-K threshold, the adversarial training phase requires the simultaneous processing of both the original and adversarial graphs. This causes memory and computational costs to increase linearly with the adversarial perturbation rate p. In large-scale graphs, concatenating adjacency matrices and features significantly increases resource overhead. As the perturbation rate increases, the robustness benefits of AARS decrease with increasing graph complexity, indicating that its defense ability is nonlinearly linked to the graph semantic density and adversarial strength, making it difficult to adapt to locally heterogeneous structures.

To address these issues, future studies could restrict the attention-filtering range to k-hop local subgraphs, thereby further reducing the complexity to $O(Nk_{\max})$. During adversarial perturbation generation, Jaccard similarity-based edge filtering can be introduced to perturb only low-similarity edges, thus controlling redundant computation. Moreover, integrating a variance-gating mechanism after the noise-injection layer can attenuate the attention activations of anomalous nodes. A graph autoencoder can also automatically correct potential anomalous connections in $\mathbf{A}_{attn}$ through a reconstruction loss. Furthermore, to improve the scalability of AARS on large graphs, spectral truncation and tensor low-rank decomposition techniques can be used to maintain the overall complexity within the $O(N \log N)$ range, thereby balancing defense capability and training efficiency.

6 Conclusions

AARS combines multiview adversarial training with a dynamic sparse attention mechanism to boost the robustness of GNNs. The framework creates multiview inputs from both the original graph and adversarial examples. It uses a Top-K attention selection strategy to dynamically reduce the influence of perturbed edges while retaining critical topological information, thereby improving the performance against adversarial attacks. A dual loss function jointly enhances the local smoothness and global structural integrity. Experiments indicate that the combined optimization of adversarial training and the attention mechanism effectively balances local and global information, maintaining high classification accuracy even with a high attack budget.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by the Guizhou Provincial Key Technology R&D Program under Grant (QKHZC(2022)YB074) and Guizhou University Science and Technology Group [2024]07.

Author Contributions: The authors confirm contribution to the paper as follows: conceptualization, methodology, validation, formal analysis, investigation, original draft preparation, and visualization: Cheng Yang; review and editing of the manuscript: Xianghong Tang, Jianguang Lu, Chaobin Wang; supervision and funding acquisition: Xianghong Tang. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author, Xianghong Tang, upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hu X, Chen H, Chen H, Liu S, Li X, Zhang S, et al. Cost-sensitive GNN-based imbalanced learning for mobile social network fraud detection. IEEE Trans Comput Soc Syst. 2023;11(2):2675–90. doi:10.1109/tcss.2023.3302651.
2. Liu T, Wang Y, Ying R, Zhao H. Muse-GNN: learning unified gene representation from multimodal biological graph data. Adv Neu Inform Process Syst. 2023;36:24661–77.
3. Zhang X, Gan M. Hi-GNN: hierarchical interactive graph neural networks for auxiliary information-enhanced recommendation. Knowl Inform Syst. 2024;66(1):115–45. doi:10.1007/s10115-023-01949-9.
4. Besta M, Scheidl F, Gianinazzi L, Kwasniewski G, Klaiman S, Müller J, et al. Demystifying higher-order graph neural networks. IEEE Trans Pattern Anal Mach Intell. 2026;48(3):2544–65. doi:10.1109/tpami.2025.3637114.
5. Sinha A, Vennam S, Sharma C, Kumaraguru P. Higher order structures for graph explanations. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 39. Palo Alto, CA, USA: AAAI Press; 2025. p. 20514–21.

6. Li L, Yang W, Bai S, Ma Z. KNN-GNN: a powerful graph neural network enhanced by aggregating K-nearest neighbors in common subspace. Expert Syst Appl. 2024;253:124217.
7. Xiao S, Wang S, Dai Y, Guo W. Graph neural networks in node classification: survey and evaluation. Mach Vision Appl. 2022;33(1):4. doi:10.1007/s00138-021-01251-0.
8. Dai H, Li H, Tian T, Huang X, Wang L, Zhu J, et al. Adversarial attack on graph structured data. In: International Conference on Machine Learning. London, UK: PMLR; 2018. p. 1115–24.
9. Zügner D, Akbarnejad A, Günnemann S. Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York, NY, USA: ACM; 2018. p. 2847–56.
10. Sun L, Dou Y, Yang C, Zhang K, Wang J, Yu PS, et al. Adversarial attack and defense on graph data: a survey. IEEE Trans Knowl Data Eng. 2022;35(8):7693–711. doi:10.1109/tkde.2022.3201243.
11. Günnemann S. Graph neural networks: adversarial robustness. In: Graph neural networks: foundations, frontiers, and applications. Singapore: Springer; 2022. p. 149–76.
12. Wu H, Wang C, Tyshetskiy Y, Docherty A, Lu K, Zhu L. Adversarial examples for graph data: deep insights into attack and defense. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19; 2019 Aug 10–16; Macao, China. p. 4816–23.
13. Entezari N, Al-Sayouri SA, Darvishzadeh A, Papalexakis EE. 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; 2020 Jan 20–24; Online. p. 169–77.
14. Zhu Y, Xu W, Zhang J, Liu Q, Wu S, Wang L. Deep graph structure learning for robust representations: a survey. arXiv:2103.03036. 2021.
15. Jin W, Ma Y, Liu X, Tang X, Wang S, Tang J. Graph structure learning for robust graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York, NY, USA: ACM; 2020. p. 66–74.
16. Zhang A, Ma J. Defensevgae: defending against adversarial attacks on graph data via a variational graph autoencoder. In: International Conference on Intelligent Computing. Cham, Switzerland: Springer; 2024. p. 313–24.
17. Chen Y, Wu L, Zaki M. Iterative deep graph learning for graph neural networks: better and robust node embeddings. Adv Neural Inform Process Syst. 2020;33:19314–26.
18. Zhu D, Zhang Z, Cui P, Zhu W. Robust graph convolutional networks against adversarial attacks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York, NY, USA: ACM; 2019. p. 1399–407.
19. Tang X, Li Y, Sun Y, Yao H, Mitra P, Wang S. Transferring robustness for graph neural network against poisoning attacks. In: Proceedings of the 13th International Conference on Web Search and Data Mining; 2020 Jan 20–24; Online. p. 600–8.
20. Zhang X, Zitnik M. Gnnguard: defending graph neural networks against adversarial attacks. Adv Neural Inform Process Syst. 2020;33:9263–75. doi:10.1109/icdmw58026.2022.00096.
21. Chen J, Wu Y, Xu X, Chen Y, Zheng H, Xuan Q. Fast gradient attack on network embedding. arXiv:1809.02797. 2018.
22. Sun Y, Wang S, Tang X, Hsieh TY, Honavar V. Adversarial attacks on graph neural networks via node injections: a hierarchical reinforcement learning approach. In: Proceedings of the Web Conference 2020; 2020 Apr 20–24; Online. p. 673–83.
23. Zhang Y, Liu X, Wu M, Yan W, Yan M, Ye X, et al. Disttack: graph adversarial attacks toward distributed GNN training. In: European Conference on Parallel Processing. Cham, Switzerland: Springer; 2024. p. 302–16.
24. Zügner D, Borchert O, Akbarnejad A, Günnemann S. Adversarial attacks on graph neural networks: perturbations and their patterns. ACM Trans Know Disc Data (TKDD). 2020;14(5):1–31. doi:10.1145/3394520.
25. Dai E, Zhao T, Zhu H, Xu J, Guo Z, Liu H, et al. A comprehensive survey on trustworthy graph neural networks: privacy, robustness, fairness, and explainability. Mach Intell Res. 2024;21(6):1011–61.
26. Xu K, Hu W, Leskovec J, Jegelka S. How powerful are graph neural networks? arXiv:1810.00826. 2019.

27. Xu K, Chen H, Liu S, Chen PY, Weng TW, Hong M, et al. Topology attack and defense for graph neural networks: an optimization perspective. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. Palo Alto, CA, USA: AAAI Press; 2019. p. 3961–7.
28. Gosch L, Sturm D, Geisler S, Günnemann S. Revisiting robustness in graph machine learning. arXiv:2305.00851. 2023.
29. Dong G, Tang M, Wang Z, Gao J, Guo S, Cai L, et al. Graph neural networks in IoT: a survey. ACM Trans Sens Netw. 2023;19(2):1–50. doi:10.1145/3565973.
30. Kong X, Song Z, Ye X, Jiao J, Qi H, Liu X. GRID: graph-based robust intrusion detection solution for industrial IoT networks. IEEE Inter Things J. 2025;12(14):26646–59.
31. Giraldo JH, Skianis K, Bouwmans T, Malliaros FD. On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. New York, NY, USA: ACM; 2023. p. 566–76.
32. Zhang C, Zhou L, Xu X, Wu J, Liu Z. Adversarial attacks of vision tasks in the past 10 years: a survey. ACM Comput Surv. 2025;58(2):1–42. doi:10.1145/3743126.
33. Zhai Z, Li P, Feng S. State of the art on adversarial attacks and defenses in graphs. Neural Comput Appl. 2023;35(26):18851–72. doi:10.1007/s00521-023-08839-9.
34. Zhang J, Hong Y, Cheng D, Zhang L, Zhao Q. Defending adversarial attacks in Graph Neural Networks via tensor enhancement. Pattern Recognit. 2025;158:110954. doi:10.1016/j.patcog.2024.110954.
35. Wang G, Ying R, Huang J, Leskovec J. Multi-hop attention graph neural networks. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21; 2021 Aug 19–26; Online. p. 3089–96.
36. Gu Y, Sun H, Wang Y, Shi H. Improve robustness of graph neural networks: multi-hop neighbors meet homophily-based truncation defense. In: 2023 International Joint Conference on Neural Networks (IJCNN). Piscataway, NJ, USA: IEEE; 2023. p. 1–8.
37. Ennadir S, Abbahaddou Y, Lutzeyer JF, Vazirgiannis M, Boström H. A simple and yet fairly effective defense for graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 38. Palo Alto, CA, USA: AAAI Press; 2024. p. 21063–71.
38. Geisler S, Schmidt T, Şirin H, Zügner D, Bojchevski A, Günnemann S. Robustness of graph neural networks at scale. Adv Neural Inform Process Syst. 2021;34:7637–49. doi:10.52202/079017-1554.
39. Fofanah AJ, Chen D, Wen L, Zhang S. Addressing imbalance in graph datasets: introducing gate-GNN with graph ensemble weight attention and transfer learning for enhanced node classification. Expert Syst Appl. 2024;255:124602. doi:10.1016/j.eswa.2024.124602.
40. Fofanah AJ, Leigh AO. EATSA-GNN: edge-aware and two-stage attention for enhancing graph neural networks based on teacher–student mechanisms for graph node classification. Neurocomputing. 2025;612:128686. doi:10.1016/j.neucom.2024.128686.
41. Huang J, Du L, Chen X, Fu Q, Han S, Zhang D. Robust mid-pass filtering graph convolutional networks. In: Proceedings of the ACM Web Conference 2023; New York, NY, USA: ACM; 2023. p. 328–38.
42. Chen H, Zhou X, Zhang J, Wang H. L2M-GCN: a new framework for learning robust GCN against structural attacks. Neurocomputing. 2025;636:129962. doi:10.1016/j.neucom.2025.129962.
43. Jia Y, Zou D, Wang H, Jin H. Enhancing node-level adversarial defenses by Lipschitz regularization of graph neural networks. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM; 2023. p. 951–63.
44. Zhou C, Peng Y, Huang W, Miao X, Cao Y, Wang X, et al. A dynamic ensemble learning model for robust Graph Neural Networks. Neural Netw. 2025;191(2):107810. doi:10.1016/j.neunet.2025.107810.