A Dual Robust Graph Neural Network Against Graph Adversarial Attacks

Qian Tao[a,b], Jianpeng Liao[a], Enze Zhang[a], Lusi Li[c,*]

[a]*School of Software, South China University of Technology, Guangzhou, Guangdong, 510006, China*
[b]*Pazhou Lab, Guangzhou, Guangdong, 510006, China*
[c]*Department of Computer Science, Old Dominion University, Norfolk, VA, 23529, USA*

Abstract

Graph Neural Networks (GNNs) have gained widespread usage and achieved remarkable success in various real-world applications. Nevertheless, recent studies reveal the vulnerability of GNNs to graph adversarial attacks that fool them by modifying graph structure. This vulnerability undermines the robustness of GNNs and poses significant security and privacy risks across various applications. Hence, it is crucial to develop robust GNN models that can effectively defend against such attacks. One simple approach is to remodel the graph. However, most existing methods cannot fully preserve the similarity relationship among the original nodes while learning the node representation required for reweighting the edges. Furthermore, they lack supervision information regarding adversarial perturbations, hampering their ability to recognize adversarial edges. To address these limitations, we propose a novel Dual Robust Graph Neural Network (DualRGNN) against graph adversarial attacks. DualRGNN first incorporates a node-similarity-preserving graph refining (SPGR) module to prune and refine the graph based on the learned node representations, which contain the original nodes' similarity relationships, weakening the poisoning of graph adversarial attacks on graph data. DualRGNN then employs an adversarial-supervised graph attention (ASGAT) network to enhance the model's capability in identifying adversarial edges by treating these edges as supervised signals. Through extensive experiments conducted on four benchmark datasets, DualRGNN has demonstrated remarkable robustness against various graph adversarial attacks.

Keywords: Graph neural network, graph adversarial attacks, dual robustness, graph attention network.

1. Introduction

Over the past few years, graph neural networks (GNNs) have attracted much attention because of their impressive ability to effectively deal with graph-structured data.

*Corresponding author.
Email addresses: taoqian@scut.edu.cn (Qian Tao), sejianpengliao@mail.scut.edu.cn (Jianpeng Liao), francis_enze_zhang@foxmail.com (Enze Zhang), lusili@cs.odu.edu (Lusi Li)

GNNs have achieved remarkable success in a variety of domains, including visual identity (Chen et al., 2019; Majumdar, 2020; Liu et al., 2023; Gou et al., 2023; Dornaika et al., 2023), recommendation systems (Song et al., 2022; Yang et al., 2022; Zhang et al., 2023; Zhang and Wang, 2023), natural language processing (Yin et al., 2020; Yu et al., 2021; Zhao et al., 2022), link prediction (Dai et al., 2022b; Salha-Galvan et al., 2022; Han et al., 2023; Zhang and Bai, 2023), etc. The success of GNNs can be attributed to the graph message propagation mechanism (Wu et al., 2021). In this mechanism, each node aggregates information from its local neighborhood in each layer at each iteration. This iterative process enables the model to eventually generate embeddings that not only capture rich node features but also encapsulate valuable topological structure information. While the information aggregation from neighboring nodes is powerful in embedding learning, the manner in which GNNs exchange this information between nodes introduces a potential risk.

Recent studies have shown that GNNs are vulnerable to adversarial attacks (Dai et al., 2018; Zügner et al., 2018; Zügner and Günnemann, 2019; Wu et al., 2019b). These attacks involve introducing subtle or carefully crafted perturbations to the graph-structured data, which may go unnoticed but have a significant detrimental impact on the performance of GNNs. The lack of robustness in GNN models can lead to severe consequences for some security and privacy-related applications. For instance, in financial credit system, fraudsters can make fraudulent transactions with normal customers to evade the high-risk customer detection models. As illustrated in Figure 1, originally the fraudster was classified by the GNN model as a high-risk customer. After making fraudulent transactions with some normal customers, the GNN model misclassifies them as a normal user. In this situation, the lack of the robustness of GNN model can lead to financial fraud and other fatal consequences. Consequently, studying robust GNN models to resist graph adversarial attacks is of great significance. Although attackers have various means to modify clean graph data, such as perturbing node features and modifying graph structure, recent research (Zügner and Günnemann, 2019; Xu et al., 2020; Finkelshtein et al., 2022) has shown that most of the existing graph adversarial attacks focus on altering the graph structure, especially by adding or rewiring edges to degrade the performance of GNNs, that is poisoning adversarial attacks. In this scenario, the graph data undergoes perturbations due to the modified graph structure and is then used to train the GNN models.

A simple solution to improve the robustness of GNNs against graph adversarial attacks is to reweight the edges and reconstruct the graph by learning the similarity relationships between nodes. Most existing methods recalculate edge weights using either the original node features or node representations learned from GNNs and their variants, such as graph convolutional networks (GCNs). For example, Jaccard-GCN (Wu et al., 2019b) modeled edge weights based on the Jaccard similarity of the original node features, while GNNGuard (Zhang and Zitnik, 2020) utilized cosine similarity to calculate edge weights at each layer. However, the original node features typically lack graph structure information and may contain redundant or noisy data. Relying solely on these features to model edge weights can lead to mistakenly removing normal edges and inaccurately reflecting the graph's topological structure. In an attempt to address this issue, GRCN (Yu et al., 2020) utilized supervised GCNs to learn node representations and adjusted edge weights based on the inner product of these learned

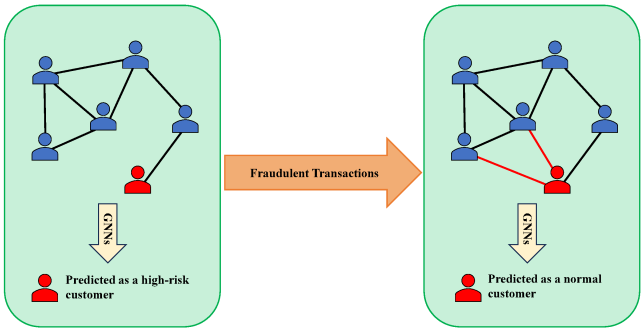


Figure 1: An example of adversarial attack on financial credit. The goal of the GNN is to predict the risk of the customers. Here the blue and red persons represents normal and high-risk customers. The high-risk customer aims to change the prediction of GNN by making fraudulent transactions with normal customers.

representations. Nonetheless, supervised GNNs are prone to interference from poisoning graphs in adversarial attacks, resulting in poor-quality learned representations.

To mitigate the drawbacks, some methods have employed GCNs to learn node representations and reconstruct the graph in an unsupervised manner, effectively circumventing issues arising from the use of original features and supervised learning. For instance, STABLE (Li et al., 2022) used unsupervised GCNs to learn node representations that capture substantial graph structure information, demonstrating the insensitivity to structural perturbations for refining the graph structure. However, a study (Jin et al., 2021) has shown that the neighborhood aggregation process of vanilla GCNs tends to preserve graph structural similarity rather than node similarity in graph representation learning. The graph structure is easily damaged by poisoning attacks, making it unreliable to only preserve graph structure similarity in GCNs. This characteristic of GCNs often destroys the node similarity relationships in the original feature space. The aggregation process smoothes the node representations, making them between node pairs on each edge more similar while causing representations of nodes without edges to become more dissimilar. Consequently, there are significant distributional differences between the learned representations and the original features, ultimately leading to a failure to maintain the node similarity of the original feature space. This misalignment between node representations and the original graph structure poses significant challenges for graph remodeling.

In addition, most existing robust GNN models either utilize extremely limited labeled samples to constrain learning node representations (Zhu et al., 2019; Zhang and Zitnik, 2020; Li et al., 2022), or rely on the intrinsic characteristics of clean real-world graphs to guide model learning, such as smoothness, sparseness, low-rank, and others. However, they lack effective utilization of adversarial information (Jin et al., 2020b). As a result, these GNN models lack the necessary supervision for recognizing and handling adversarial edges effectively.

To tackle the above challenges, we propose a novel Dual Robust Graph Neural Network (DualRGNN) as a defense mechanism against graph adversarial attacks. DualRGNN comprises two main components: a node-similarity-preserving graph refining

(SPGR) module and an adversarial-supervised graph attention (ASGAT) network. On one hand, the SPGR module aims to refine the graph structure by leveraging the learned node representations. This module can preserve the original nodes' similarity, encapsulate graph topology information, and be insensitive to perturbations, thereby mitigating the detrimental effects of graph adversarial attacks on the graph data. On the other hand, the ASGAT network employs adversarial edge information as supervised signals to train an adversarial-supervised attention mechanism. This enables the network to recognize the adversarial edges and suppress the propagation of adversarial information within the graph, thereby enhancing the model's ability to resist graph adversarial attacks. The above two modules are designed to defend against graph adversarial attacks from the perspective of graph data and models, respectively. By integrating these two modules effectively, DualRGNN can improve the robustness of GNNs against graph adversarial attacks, achieving higher levels of resilience and provides an effective defense mechanism against these attacks. The main contributions can be summarized as follows.

- We propose a novel Dual Robust Graph Neural Network (DualRGNN) against poisoning graph adversarial attacks on the graph structure, which improves the robustness of GNNs from graph structure and graph representation learning.
- We propose to refine the graph structure by learning node representations that preserve the similarity relationships among the original nodes and carry graph topology information while remaining insensitive to perturbations, to weaken the poisoning of graph adversarial attacks on the graph data.
- We present a new adversarial-supervised graph attention network that exploits adversarial edge information as supervised signals, and develop an adversarial-supervised attention mechanism to improve the model's ability to recognize adversarial edges.
- Extensive experiments on four real-world datasets demonstrate the effectiveness of DualRGNN in defending against various types of adversarial attacks. Additionally, an ablation study is performed to validate the efficacy of each component of DualRGNN.

2. Related Work

2.1. Graph Adversarial Attacks

Extensive studies have shown that traditional deep neural networks are vulnerable to adversarial attacks. GNNs transfer the key ideas of traditional deep neural networks to graph data processing. In graph-structured data, nodes are interdependent and correlate to each other. GNNs perform neighborhood aggregation through a graph message propagation mechanism to learn node representation. When a node in the graph is subjected by adversarial attacks, the adversarial information will contaminate other nodes along with the graph message propagation process, greatly reducing the performance of GNNs. Therefore, GNNs are more vulnerable to persecution from adversarial attacks (Dai et al., 2018; Zügner et al., 2018; Zügner and Günnemann, 2019; Wu et al.,

2019b). The goal of graph adversarial attacks is to cause GNNs to misclassify graph nodes by perturbing the graph structure or node features.

According to the time of attack occurrence, graph adversarial attacks can be divided into poisoning attacks and evasion attacks (Jin et al., 2020a). Poisoning attacks take place during training, where the attacker manipulates the training data. Evasion attacks perturb the graph data during testing, where GNN models are trained on clean graphs and then tested on perturbed data. In our work, we mainly focus on defending against the poisoning adversarial attacks. Poisoning adversarial attacks can be divided into two categories: targeted attacks and non-targeted attacks. In a targeted attack, the attacker aims to cause the GNNs model to make incorrect predictions on the target nodes. While for non-targeted attacks, the attackers focus on reducing the overall performance of GNN models on all test nodes. One of the most typical poisoning attack methods is Nettack (Zügner et al., 2018), which generates unnoticeable perturbations by modifying graph structure and node attributes, and simultaneously preserving important data characteristics such as degree distribution. Metattack (Zügner and Günnemann, 2019) regarded graph structure as a hyperparameter, introducing meta-learning into graph adversarial attacks and optimizing the algorithm based on meta gradients. In our experiments, we will utilize Metattack and Nettack to verify the performance of our method in resisting various graph adversarial attacks.

2.2. Robust Graph Neural Networks

The vulnerability of GNNs to graph adversarial attacks has led to extensive research on robust graph neural networks (RGNN). Ding et al. (2018) first introduced generative transitional networks (GANs) into GNNs, generating fake samples in low-density areas to reduce the influence of samples nearby, thereby improving the robustness of GNNs. However, the introduction of GANs also brings significant computation overhead. Jin and Zhang (2019) proposed to perturb the embedding representations of hidden layers in GCNs to indirectly generate perturbations to the graph, which can implicitly enforce the robustness of GNNs while greatly reducing computing cost. Feng et al. (2021) proposed a GraphAT, designing a graph adversarial regularizer that can maximize the divergence between the prediction of the target examples and its connected nodes when generating perturbations. However, graph adversarial training methods rely on clean graph data and cannot resist poisoning attacks. Zhu et al. (2019) provided a Robust GCN (RGCN), adopting Gaussian distributions as the hidden representations of nodes in each GCNs layer and designing a variance-based attention mechanism to absorb the adversarial attacks. By designing a penalized aggregation mechanism, Tang et al. (2020) proposed a PA-GNN, which treated the adversarial edges as supervisions of known perturbations in the clean graphs with similar domains, and transferred this ability to the target poisoned graph with meta-learning. However, the transfer-based robustness is extremely limited, and besides, a clean graph with similar domains is not always available in the real world.

Wu et al. (2019b) proposed a simple pre-processing method, checking the node pairs of each edge in the graph based on Jaccard similarity and removing those edges with low similarity scores to obtain a clean graph. Similarly, by introducing the singular value decomposition (SVD), Entezari et al. (2020) suggested reconstructing the graph with the low-rank approximation of the adjacency. However, the pre-processing

methods only process the input graph data once and are independent of the pipeline of GCNs, making it difficult to obtain a clean graph effectively. Yu et al. (2020) proposed a graph-revised convolutional network (GRCN) to revise the graph structure in each GCNs layer. Similarly, Zhang and Zitnik (2020) proposed GNNGuard, assigning higher weights to edges connecting similar nodes while pruning edges between unrelated nodes in each GNNs layer, and simultaneously introducing a layer-wise graph memory module to stabilize training. Zhang and Ma (2020) suggested utilizing variational graph autoencoders to encode the disturbed graph into latent representations to reconstruct it. Jin et al. (2020b) proposed Pro-GNN, utilizing the low rank, sparsity, and smoothness of real-world clean graphs to learn robust graph structures that conform to the above characteristics. Li et al. (2022) proposed STABLE, which refined the graph structure by learning representations that carry both feature and structure information and are insensitive to perturbations, based on unsupervised graph contrastive learning, and simultaneously designed a robust normalization mechanism to improve GCNs. Dai et al. (2022a) proposed RS-GNN to address graph denoising by learning a link predictor to down-weight noisy edges while reinforcing connections between nodes with high similarity, thereby enhancing the efficacy of message-passing within GNNs. Recently, Wang et al. (2023) proposed USER, an unsupervised framework that utilizes intrinsic graph connectivity, such as the rank of the adjacency matrix and a structural entropy-based objective function, to mitigate the adverse impacts of random perturbations.

However, most existing GCN-based robust GNNs primarily concentrate on preserving graph structure similarity, often neglecting the inherent node similarity in the original features, especially for nodes lacking direct edges between them. Consequently, refining the graph structure based on the original node similarity relationships becomes essential. Moreover, a notable limitation across most existing robust GNN models lies in their incapacity to recognize adversarial edges which are mixed with normal edges, hampering the performance of the message-passing mechanism. Hence, there is a need for supervision in reweighting edges to alleviate the influence of adversarial edges.

In this paper, we aim to refine the graph structure by learning node representations that preserve the similarity of original nodes, aiming to mitigate the detrimental effects of graph adversarial attacks on graph data. Additionally, we treat the adversarial edges as supervised signals to improve GNNs' ability to recognize and identify these edges, thereby purging the process of neighborhood aggregation to obtain cleaner node representations. By introducing these two mechanisms, the robustness to resist graph adversarial attacks will be significantly enhanced.

3. Proposed Method

3.1. System Model

In this section, we follow (Guo et al., 2022; Cinà et al., 2023) to introduce our system model according to attacker's goal, knowledge of the target system, and capability of manipulating the input data. Subsequently, we outline the defender's objectives, knowledge, and capabilities.

3.1.1. Attack Framework

In the context of graph adversarial attacks, the objective of a poisoning attack is to diminish the performance of GNN models by injecting adversarial samples into the training dataset. In our method, the attacker executes adversarial attacks by injecting adversarial edges that do not exist in the original graph datasets before the GNN model is trained. The attacker's knowledge includes all information about the architecture of the model and the original training data. The attacker's capabilities include launching various types of adversarial attacks, such as targeted attacks and non-targeted attacks. In our experiments, we consider two types of non-targeted attacks, one type of targeted attack, and random noises.

3.1.2. Defense Framework

The defender's goal is to improve the accuracy and availability of the model by resisting adversarial attacks initiated by the attacker. When the attacker tries to maximize the loss value by injecting poisons into the clean data, the defender struggles to minimize the loss under various adversarial attacks. Despite lacking access to the clean training data, the defender can refine or purify the poisoned data from the attacker. Additionally, the defender can adjust the architecture and parameters of the model, providing a robust defense against adversarial attacks.

In this paper, the proposed model plays the role of the defender, including two modules. The SPGR module refines the poisoned graph using both original node features and multiple graph structures. The ASGAT module injects the adversarial edges which are removed in SPGR module, treating them as supervised signals to train a GAT network, enabling the model to recognize the adversarial edges in the perturbed input graph. In the testing phase, the perturbed graph with refined node representations is directly fed into the GAT network. With the trained attention weights, the adversarial edges in the graph can be recognized and the propagation of perturbations can be suppressed by the defender.

3.2. Overall Architecture

The overall architecture of the proposed DualRGNN framework is shown in Fig. 2. The blue part denotes the SPGR module, and the green one represents the ASGAT network. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be an undirected graph with $X \in \mathbb{R}^{N \times d}$ as the node feature matrix, where there are N nodes and each node has a d-dimensional feature vector.

In the SPGR module, we adopt a graph contrastive learning strategy (Velickovic et al., 2019; You et al., 2020) and design four types of graph augmentations through edge remodeling, dropping, and recovering to generate correlated views for contrastive learning. Specifically, given an input perturbed graph $\mathcal{G}$ with node features X, $\mathcal{G}$ is first converted to a k-nearest neighbor (kNN) graph $\mathcal{G}^K$ by modeling original node similarity relationships. The generated kNN graph $\mathcal{G}^K$ can be regarded as the first augmented graph view for the input graph $\mathcal{G}$. Meanwhile, the input graph $\mathcal{G}$ is pre-processed by initially dropping its plausible adversarial edges to produce a relatively clean graph $\mathcal{G}^P$ as the second augmented graph view. Based on $\mathcal{G}^P$, we then randomly recover a small portion of removed edges to generate a set of perturbed augmented graph views $\mathcal{G}_m^P$ with different recovering ratios as the third type of graph views. In addition, we shuffle

$\mathcal{G}^P$ to obtain the fourth graph view $\overline{\mathcal{G}}^P$. Thus, the given graph $\mathcal{G}$ undergoes graph data augmentations to obtain four types of correlated views that contain diverse graph-structured information. To jointly preserve the original nodes' similarity and combine this information, we employ a similarity-preserved graph convolution layer with shared parameters to learn node representations from three pairs with $\mathcal{G}^K$ and each of the other three types of graphs. In the testing phase, node representation H^P can be produced from the similarity-preserved graph convolution layer acting on $\mathcal{G}^P$ and $\mathcal{G}^K$.

Once we obtain the satisfactory node representation H^P, we prune and refine the input graph $\mathcal{G}$ based on it, and obtain a robust graph $\mathcal{G}^*$ which serves as the input to ASGAT.

In the ASGAT network, we inject adversarial edges into $\mathcal{G}^*$ to obtain the perturbed graph $\tilde{\mathcal{G}}^*$ by randomly recovering a small portion of edges removed during the graph structure refining. These adversarial edges are treated as supervised signals to train the adversarial-supervised attention mechanism. We then employ the graph contrastive learning strategy between $\mathcal{G}^*$ and $\tilde{\mathcal{G}}^*$ to optimize this module. In the prediction stage, the class prediction Z can be obtained from the $\mathcal{G}^*$ using the ASGAT network. More specific designs of DualRGNN are elaborated as follows.

3.3. Node-Similarity-Preserving Graph Refining Module

The proposed DualRGNN employs an SPGR module to cleanse and refine the input graph to resist poisoning attacks from the perspective of graph data. To learn robust representations from graph-structured data, we adopt a graph contrastive learning strategy (Velickovic et al., 2019; You et al., 2020) to capture diverse information from different types of graph augmentations in a contrastive learning manner. Specifically, we first generate four types of graph data augmentations, then perform graph convolution on them to obtain graph representations through contrastive learning, and refine the input graph structure with these representations.

3.3.1. Graph augmentation generation

Creating augmented graph views is the prerequisite for graph contrastive learning, as diverse augmented views of a graph can offer various contexts for this learning process. In the SPGR module, we aim to refine the graph structure by leveraging the acquired node representations. There representations maintain the similarity of the original nodes while incorporating extensive graph topology information, making them resilient to disturbances.

To capture the similarity information of the original nodes, we introduce a kNN graph, which models the node similarity relationships in the initial feature space. This graph can be treated as the first type of augmented graph. For the given input graph $\mathcal{G}$ containing N nodes, we employ the kNN graph construction algorithm using cosine similarity to compute each similarity score C_{ij} between the i-th and j-th nodes in the original feature space. The kNN graph is created by selecting the k (where $k \ll N$) nearest neighbors to each target node. We denote this graph as $\mathcal{G}^K$, and its adjacency matrix is represented as A^K.

$$C_{ij} = \mathrm{sim}_{\mathrm{Cos}}\left(x_i, x_j\right), \tag{1}$$

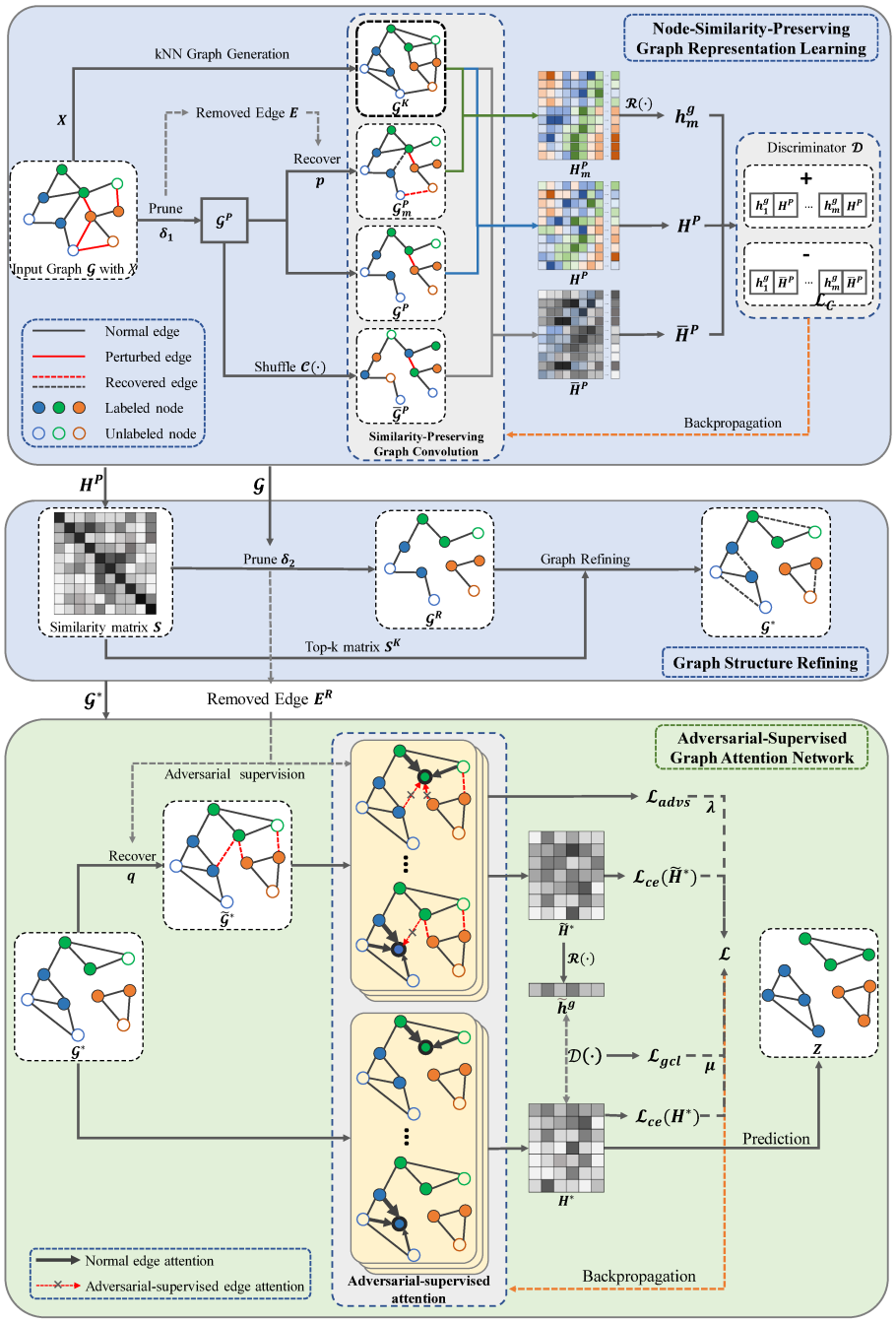


Figure 2: Overview of the proposed DualRGNN framework. DualRGNN mainly consists of two components: a node-similarity-preserving graph refining module (in the blue part), and an adversarial-supervised graph attention network (in the green part).

$$A_{ij}^{K} = \begin{cases} 1, & \text{if } x_j \in \text{knear}\,(x_i) \\ 0, & \text{else,} \end{cases} \tag{2}$$

where x_i is the feature vector of i-th node, and knear (x_i) denotes the k-nearest neighbour nodes of node x_i and is determined by C_{ij}.

In order to obtain a relatively clean graph, following the previous work (Wu et al., 2019b), we preprocess the input perturbed graph by roughly removing adversarial edges that usually connect two nodes with low similarity. The purged graph can be denoted as $\mathcal{G}^P$ as the second type of augmented graph. Specifically, we assess the similarity score between paired nodes of each edge in the input graph based on Jaccard similarity, which is shown as follows:

$$J_{ij} = \text{sim}_{\text{Jac}}\left(x_i, x_j\right), \tag{3}$$

where $\text{sim}_{\text{Jac}}(\cdot)$ is a similarity measure function. We prune the input graph based on the similarity matrix J by removing the edges whose similarity score is below a predefined threshold δ_1, which is defined as:

$$A_{ij}^{P} = \begin{cases} 1, & \text{if } J_{ij} \geq \delta_1 \\ 0, & \text{if } J_{ij} < \delta_1, \end{cases} \tag{4}$$

where A^P is the adjacency matrix of $\mathcal{G}^P$. The removed edge can be denoted by a matrix E, where $E_{ij} = 1$ denotes that the edge between x_i and x_j has been removed. It can be noted that the adjacency matrices A, A^P and E satisfy the formula as:

$$A = A^P + E. \tag{5}$$

Inspired by the previous work (Li et al., 2022), we introduce several perturbed augmented graph views with slight adversarial perturbations into graph contrastive learning, to learn the node representations that are insensitive to perturbations. We generate the perturbed augmented graph views by randomly recovering a small portion of removed edges as the third type of augmented graph. Formally, we first sample a random masking matrix $M^p \in \{0,1\}^{N\times N} \sim \mathcal{B}(1-p)$ from a Bernoulli distribution, where p is a hyper-parameter that controls the recovery ratio. Then we recover a small portion of edges into $\mathcal{G}^P$ based on the masking matrix M^p to obtain the perturbed augmented graph view $\mathcal{G}_m^P$, which can be formalized as:

$$A_m^P = A^P + E \bigodot M^p, \tag{6}$$

where $\bigodot$ represents Hadamard product. A_m^P is the adjacency matrix of $\mathcal{G}_m^P$. Considering the trade-off between performance and computation overhead, we adopted two perturbed augmented graph views for implementation.

Following the previous work (Velickovic et al., 2019; Li et al., 2022), we introduce a corruption function $C(\cdot)$ to generate a negative graph as the fourth type. The negative graph $\overline{\mathcal{G}}^P$ is obtained by randomly shuffling the feature rows of $\mathcal{G}^P$ and keeping the graph topology. That means $\mathcal{G}^P$ and $\overline{\mathcal{G}}^P$ have the same topology structure and different node orders.

In summary, the augmented graphs $\mathcal{G}^K$, $\mathcal{G}^P$, and $\mathcal{G}_m^P$ are positive instances with distinct adjacency matrices while sharing identical node feature representations with the original graph $\mathcal{G}$. Notably, $\mathcal{G}^K$ is generated with kNN graph construction algorithm using cosine similarity between the features of each node and its neighbors, capturing node-wise similarity information. In the similarity-preserving graph convolution, $\mathcal{G}^K$ is used to simultaneously aggregate the information from the nodes with higher similarities, which is performed in Eq.(8). Note that when generating $\mathcal{G}^K$, the adjacency matrix of $\mathcal{G}$ is not used. In a clean graph, nodes with low feature similarity are usually not connected by edges. If an edge connects a pair of nodes with low similarity, it is likely to be an adversarial edge. Therefore, we generate a relatively clean graph $\mathcal{G}^P$ by removing the edges that connects two nodes with low similarity. $\mathcal{G}_m^P$ is obtained by recovering a small portion of removed edges in $\mathcal{G}^K$, and this process can be regarded as injecting slight attacks. Finally, we randomly shuffle the feature rows of $\mathcal{G}^K$ to obtain $\overline{\mathcal{G}}^P$ as the negative augmented graph in our strategy of contrastive learning.

Subsequently, we introduce a similarity-preserved graph convolution layer into the SPGR module to learn the node representations that can preserve the original nodes' similarity. More details of the similarity-preserved graph convolution will be described as follows.

3.3.2. Similarity-preserved graph convolution

Similarity-preserving graph convolution can consider both the graph structure information and the original node similarity relationship when learning node representations, aggregating neighborhood feature information from both the graph structure and the original node similarity relationship, and thereby the learned node representations can not only carry graph structure information but also preserve the original node similarity.

For the similarity-preserved graph convolution, the convolution operation in the l-th layer ($l \in \mathbb{Z}^+$) is formalized as follows:

$$H^{(l)} = \sigma\left(P^{(l)} H^{(l-1)} W^{(l)}\right), \tag{7}$$

where $H^{(l-1)} \in \mathbb{R}^{N \times d^{(l-1)}}$ is the input node feature matrix from a previous layer and $H^{(0)} = X$. $\sigma(\cdot)$ is an activation function. $W^{(l)}$ is the weight to be trained in the l-th layer. $P^{(l)}$ is the graph Laplacian matrix in this layer.

Unlike the vanilla GCNs, the graph Laplacian matrix $P^{(l)}$ integrates both the propagation information of the original graph and the kNN graph, enabling it to aggregate the neighborhood information from these two graphs, which is defined as:

$$P^{(l)} = s^{(l)} * D^{P^{-\frac{1}{2}}} A^P D^{P^{-\frac{1}{2}}} + \left(1 - s^{(l)}\right) * D^{K^{-\frac{1}{2}}} A^K D^{K^{-\frac{1}{2}}}, \tag{8}$$

where D^P and D^K are the node degree matrix of graph $\mathcal{G}^P$ and $\mathcal{G}^K$, respectively. $s^{(l)}$ is the weight vector that adaptively balances the neighborhood information from $\mathcal{G}^P$ and $\mathcal{G}^K$. $s * A$ denotes the multiplication of the i-th element of vector s and the i-th row of matrix A. Therefore, $s^{(l)}$ allows nodes to adaptively integrate information from $\mathcal{G}^P$ and $\mathcal{G}^K$, as each node can have different weight scores. In order to reduce the number

of parameters in $s^{(l)}$, we model it as a mapping of node representations, which can be formalized as:

$$s^{(l)} = \text{sigmoid}\left(H^{(l-1)}W_s^{(l)} + b^{(l)}\right), \tag{9}$$

where $W_s^{(l)} \in \mathbb{R}^{d^{(l-1)}\times 1}$ and $b^{(l)} \in \mathbb{R}$ are the parameters to be trained to map $H^{(l-1)}$ to $s^{(l)}$. sigmoid$(\cdot)$ is the activation function. Through this method, the number of parameters of $s^{(l)}$ has been cut down from N to $d^{(l-1)} + 1$, thereby reducing the computation cost of the model.

We adopt the similarity-preserved graph convolution to perform graph representation learning for the purged graph $\mathcal{G}^P$, perturbed augmented graph $\mathcal{G}_m^P$, and the negative graph $\overline{\mathcal{G}}^P$, combining with the kNN graph $\mathcal{G}^K$, and obtain the corresponding node representation matrices H^P, H_m^P and $\overline{H}^P$.

The proposed SPGR module is based on graph contrastive learning, which optimizes the network by maximizing the mutual information between the global representations of the perturbed augmented graph and the local representations of the purged graph. The global-local contrastive learning is conducive to learning node representations insensitive to perturbations.

To obtain the global representations of the augmented graph, we introduce a readout function $\mathcal{R}(\cdot)$ that is implemented with a global average pooling function (Velickovic et al., 2019) as:

$$h_m^g = \mathcal{R}\left(H_m^P\right) = \text{sigmoid}\left(\frac{1}{N}\sum_{i=1}^{N} H_{m\,i}^P\right), \tag{10}$$

where $H_{m\,i}^P$ denotes the i-th row of H_m^P, that is the representation corresponding to the i-th node in $\mathcal{G}_m^P$.

We introduce a discriminator $\mathcal{D}$ to distinguish positive samples pairs and negative samples pairs, as well as output the probability score of the sample pair from the joint distribution, which can be implemented using Bilinear operations as follows:

$$\mathcal{D}\left({H^P}_i, h_1^g\right) = \sigma\left({H^P}_i^{\top} W^D h_1^g\right), \tag{11}$$

where ${H^P}_i$ is the i-th row of H^P. W^D is a scoring matrix to be trained.

To maximize the mutual information, a binary cross-entropy loss between positive samples and negative samples is adopted as the objective function of the SPGR module, which can be formalized as follows:

$$\mathcal{L}_C(\Theta) = -\frac{1}{2N}\sum_{i=1}^{N}\sum_{m=1}^{M}[\log\left(\mathcal{D}\left({H^P}_i, h_m^g\right)\right) + \log\left(1 - \mathcal{D}\left({\bar{H}^P}_i, h_m^g\right)\right)], \tag{12}$$

where $\Theta = \{W^{(l)}, W_s^{(l)}, W^D, b^{(l)}\}$ is a set containing all of the trainable parameters. $\log(\cdot)$ denotes the logarithmic operation, and M is the number of perturbed augmented graph views adopted in the SPGR module.

3.3.3. Graph structure refining

The key idea of graph structure refining is to remove adversarial edges while adding more useful edges connecting nodes with higher similarity, simultaneously.

Previous work (Zügner and Günnemann, 2019; Li et al., 2022) has shown that nodes with higher degrees are more robust and reliable compared than those with lower degrees. Due to the graph message propagation mechanism in GNNs, nodes with only a few neighbors are more vulnerable to graph adversarial attacks than nodes with a large number of neighbors. Therefore, injecting more useful edges, which usually connect two high-similarity nodes that are more likely in the same class (Li et al., 2022), into the graph through graph structure refining can diminish the impact of graph adversarial attacks on graph data.

In the prediction stage, the node representation can be obtained through the similarity-preserved graph convolution layer acting on $\mathcal{G}^P$ combining with $\mathcal{G}^K$. We adopt a cosine similarity function $\mathrm{sim}_{\mathrm{Cos}}(\cdot)$ to measure the nodes similarity as:

$$S_{ij} = \mathrm{sim}_{\mathrm{Cos}}\left(h_i, h_j\right), \tag{13}$$

where h_i and h_j are the representations of the i-th and j-th nodes, respectively. Then, based on the similarity matrix S, we prune the edges whose similarity score are below a predefined threshold δ_2 from the input graph $\mathcal{G}$, and obtain a clean graph $\mathcal{G}^R$:

$$A^R_{ij} = \begin{cases} 1, & S_{ij} \geq \delta_2 \text{ and } A_{ij} = 1 \\ 0, & S_{ij} < \delta_2 \end{cases} \tag{14}$$

where A^R is the adjacency matrix of $\mathcal{G}^R$. The removed edges can be denoted by a matrix E^R.

Then we construct a Top-K matrix S^K based on the similarity matrix S to insert useful edges into the graph $\mathcal{G}^R$. For each node in $\mathcal{G}^R$, we connect it with K neighbors most similar to it. In this way, we can obtain the final graph $\mathcal{G}^*$ with adjacency matrix A^*:

$$A^* = A^R + S^K. \tag{15}$$

The entire workflow of the SPGR module is summarized in Algorithm 1. Once we obtain the final graph $\mathcal{G}^*$, we can perform graph representation learning based on it.

3.4. Adversarial-Supervised Graph Attention Network

Most existing robust GNN models are short of effectively exploiting adversarial information, which leads to the lack of ability to directly recognize adversarial edges. Based on the transfer learning, Tang et al. (2020) treated the adversarial edges as supervisions of known perturbations in the clean graph with similar domains to train a robust GNN model and transferred it to the target graph in the way of meta-optimization. However, in the real world, clean graphs with similar domains are not always available, and the robustness achieved through transfer learning is extremely limited. To tackle this defect, our DualRGNN employs the ASGAT network explicitly treating adversarial edges as supervision signals directly on the target graph to enhance the model's ability to recognize adversarial edges.

3.4.1. Adversarial-supervised attention mechanism

The core of the ASGAT network is an adversarial-supervised attention mechanism that treats adversarial edges as supervised signals to constrain model training.

Algorithm 1: The algorithm of the SPGR module.

Input: Input perturbed graph $\mathcal{G}$ with node features X and adjacency matrix A.
Output: Robust Graph $\mathcal{G}^*$.

Initialize the parameters $W^{(l)}$, $W_s^{(l)}$, W^D, and $b^{(l)}$;
$\mathcal{G}^K \leftarrow$ Create kNN graph based on Eq.(1, 2);
$J \leftarrow$ Check similarity score between paired nodes according to Eq.(3);
$\mathcal{G}^P \leftarrow$ Remove adversarial edges using Eq.(4);
for $m = 1 \rightarrow M$ **do**
    $M^p \leftarrow$ Sample a random mask with ratio p;
    $\mathcal{G}_m^P \leftarrow$ Recover edges to generate perturbed augmented graph views according to Eq.(6);
end
$\overline{\mathcal{G}}^P \leftarrow C(\mathcal{G}^P)$;
for $i = 1 \rightarrow epochs$ **do**
    $H^P, H_m^P, \overline{H}^P \leftarrow$ Perform similarity-preserved graph convolution according to Eq.(7, 8, 9);
    $h_m^g \leftarrow \mathcal{R}\left(H_m^P\right)$;
    Calculate $\mathcal{L}_C$ according to Eq.(12);
    Perform back-propagation to update parameters;
end
`// Prediction stage`
$H^P \leftarrow$ Perform similarity-preserved graph convolution according to Eq.(7, 8, 9);
$S \leftarrow$ Check similarity between nodes via Eq.(13);
$\mathcal{G}^R \leftarrow$ Prune edges of $\mathcal{G}$ using Eq.(14);
$\mathcal{G}^* \leftarrow$ Insert useful edges into $\mathcal{G}$ via Eq.(15).

Firstly, we inject adversarial edges into $\mathcal{G}^*$ by randomly recovering a small portion of edges removed in the graph structure refining stage to obtain the perturbed graph $\tilde{\mathcal{G}}^*$ whose adjacency matrix is represented as $\tilde{A}^*$. In the implementation, we sample a random masking matrix $M^q \in \{0,1\}^{N\times N} \sim \mathcal{B}(1-q)$ from a Bernoulli distribution to recover a small portion of removed edges, where q is a hyper-parameter that controls the recovery ratio, that is,

$$\tilde{A}^* = A^* + E^R \bigodot M^q, \tag{16}$$

The designed adversarial-supervised attention mechanism adopts an implementation similar to GATs (Velickovic et al., 2018) whose attention weights are calculated as follows:

$$a_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\alpha^T \left[Wx_i \| Wx_j\right]\right)\right)}{\sum_{x_k\in\mathcal{N}_{x_i}} \exp\left(\text{LeakyReLU}\left(\alpha^T \left[Wx_i \| Wx_k\right]\right)\right)}, \tag{17}$$

where $\alpha \in \mathbb{R}^{2d\times 1}$ and W are the mapping matrix and weight matrix to be trained, respectively. $\exp(\cdot)$ denotes the exponential function, and $\text{LeakyReLU}(\cdot)$ is an activation function. $\mathcal{N}_{x_i}$ denotes the set of neighboring nodes of x_i, and $\|$ represents the concatenation operation. The attention weight a_{ij} means the importance of node x_j in the process of neighborhood aggregation to update the representation of x_i. Then we perform neighborhood aggregation based on this attention weight, which can be formalized as follows:

$$\tilde{x}_i = \sigma\left(\sum_{x_j\in\mathcal{N}_{x_i}} a_{ij}x_j\right), \tag{18}$$

However, the vanilla graph attention mechanism does not have the ability to recognize adversarial edges because it lacks supervision of adversarial information and is not robust to graph adversarial attacks. We develop the adversarial-supervised attention mechanism, which treats adversarial edges as supervision signals to constrain the model to allocate higher attention to normal edges and lower attention to adversarial edges, so as to suppress the propagation of adversarial perturbations and enhance the model's ability to recognize adversarial edges.

The optimization objectives of the adversarial-supervised attention mechanism are as follows:

$$\mathcal{L}_{advs} = -\min\left(\eta, \mathop{\mathbb{E}}_{e_{ij}\in\mathcal{E}\setminus\mathcal{P}} S - \mathop{\mathbb{E}}_{e_{ij}\in\mathcal{P}} S\right), \tag{19}$$

where η is a predefined hyperparameter, and $\mathcal{E}\setminus\mathcal{P}$ and $\mathcal{P}$ respectively denote the set of normal edges and adversarial edges. $\mathop{\mathbb{E}}_{e_{ij}\in\mathcal{E}\setminus\mathcal{P}} S$ and $\mathop{\mathbb{E}}_{e_{ij}\in\mathcal{P}} S$ represent the expected attention weights for normal and adversarial edges, respectively, which can be calculated as follows:

$$\mathop{\mathbb{E}}_{e_{ij}\in\mathcal{E}\setminus\mathcal{P}} S = \frac{1}{|\mathcal{E}\setminus\mathcal{P}|}\sum_{e_{ij}\in\mathcal{E}\setminus\mathcal{P}} a_{ij}, \tag{20}$$

$$\mathop{\mathbb{E}}_{e_{ij}\in\mathcal{P}} S = \frac{1}{|\mathcal{P}|}\sum_{e_{ij}\in\mathcal{P}} a_{ij}. \tag{21}$$

However, it is not feasible to directly optimize the adversarial-supervised attention mechanism. On the one hand, there is a difference in the distribution between the

perturbed graph $\tilde{\mathcal{G}}^*$ injected with adversarial information and the original graph $\mathcal{G}^*$, and the adversarial-supervised attention mechanism trained on the perturbed graph $\tilde{\mathcal{G}}^*$ cannot be directly used for prediction on the original graph $\mathcal{G}^*$. On the other hand, the perturbed graph $\tilde{\mathcal{G}}^*$ is disturbed by adversarial information, so performing class prediction directly on the perturbed graph $\tilde{\mathcal{G}}^*$ is not the optimal solution. To address this issue, we optimize the adversarial-supervised attention mechanism based on graph contrastive learning.

3.4.2. Contrastive-learning-based optimization

We optimize the ASGAT network based on the contrastive learning between the global representations of the perturbed graph and the local representations of the original graph. The motivation is that graph adversarial attacks need to ensure that the attacks are globally imperceptible when poisoning the graph structure. This implies that graph adversarial attacks may cause significant damage to the local part of the graph, while without affecting the global graph.

In the ASGAT network, the perturbed graph $\tilde{\mathcal{G}}^*$ with a small number of adversarial edges is regarded as an augmented graph. For the node representation $\widetilde{H}^*$ of $\tilde{\mathcal{G}}^*$, we adopt the readout function $\mathcal{R}(\cdot)$ to obtain its global representation $\widetilde{h}^g$, which is defined as Eq.(10).

Then we optimize the model by maximizing the mutual information between the global representation of the perturbed graph and the local representation of the original graph. The objective function is defined as follows:

$$\mathcal{L}_{gcl} = -\frac{1}{N}\sum_{i=1}^{N}\left[\log\left(\mathcal{D}\left({H^*}_i, \tilde{h}^g\right)\right)\right], \tag{22}$$

where $\mathcal{D}$ is the discriminator that defined as Eq.(11).

The classification loss adopts in the ASGAT network is the cross-entropy loss defined as follows:

$$\mathcal{L}_{ce} = -\sum_{i\in L}\sum_{j=1}^{c} Y_{ij}\ln Z_{ij}, \tag{23}$$

where Y denotes the label matrix, and L is the set of labeled samples. Z is the class prediction result, which can be obtained by performing a softmax$(\cdot)$ for the node representations.

The objective function of the ASGAT network is obtained by linearly combining the adversarial-supervised attention loss, contrastive learning loss, and cross-entropy loss, that is,

$$\mathcal{L}(\Phi) = \mathcal{L}_{ce} + \lambda\mathcal{L}_{advs} + \mu\mathcal{L}_{gcl}, \tag{24}$$

where $\Phi = \{W, W^D, \alpha\}$ is a set containing all trainable parameters. λ and μ are the trade-off parameters.

In the prediction stage, the class prediction result can be obtained from the original graph $\mathcal{G}^*$ based on the ASGAT network. The entire algorithm of the ASGAT network is summarized in Algorithm 2.

Algorithm 2: The algorithm of the ASGAT network.

Input: $\mathcal{G}^*$ obtained from the SPGR module in Algorithm 1.
Output: The model parameters after training, the class prediction Z.

Initialize the parameters W, W^D, α;
while *not converges* **do**
  $M^q \leftarrow$ Sample a random mask with ratio q;
  $\tilde{\mathcal{G}}^* \leftarrow$ Inject adversarial edges into $\mathcal{G}^*$ via Eq.(16);
  $H^*, \widetilde{H}^* \leftarrow$ Perform attention neighborhood aggregation according to Eqs.(17, 18);
  Calculate $\mathcal{L}_{advs}$ according to Eq.(19);
  $\tilde{h}^g \leftarrow \mathcal{R}(\widetilde{H}^*)$;
  Calculate $\mathcal{L}_{gcl}$ according to Eq.(22);
  Calculate $\mathcal{L}_{ce}$ according to Eq.(23);
  Calculate $\mathcal{L}$ according to Eq.(24);
  Perform back-propagation to update parameters;
end
$Z \leftarrow \text{softmax}(H^*)$.

3.5. Computational Complexity Analysis

3.5.1. The computational complexity of SPGR module

As shown in Algorithm 1, given the input graph $\mathcal{G}$ with N nodes, we firstly need to generate the four types of augmentations, and the time complexity of the process is $O(N^2)$. Then we train the similarity-preserving GCN model. According to (Kipf and Welling, 2017), for the l-th layer of the model, the time complexities of both forward pass and backward pass are $O(Nd^{(l-1)}d^{(l)} + N|\mathcal{E}|d^{(l)})$, where $|\mathcal{E}|$ denotes the number of edges in the graph $\mathcal{G}$. The total cost of the GCN training is $O(e_1(Nd_1^2 + N|\mathcal{E}|d_1))$, where d_1 denotes the maximum dimension of features in the GCN model and e_1 denotes the number of epochs of training. Finally, we perform similarity-preserved graph convolution and obtain the robust graph $\mathcal{G}^*$, and the total complexity should be $O(Nd_1^2 + N|\mathcal{E}|d_1 + N^2)$. Considering N is usually in the thousands and both e_1 and d_1 are in the hundreds, the order of magnitude of N and e_1d_1 are close. Therefore, the total time complexity of SPGR module is $O(e_1d_1N(d_1 + |\mathcal{E}|) + N^2)$.

3.5.2. The computational complexity of ASGAT network

As shown in Algorithm 2, the main part is a loop which trains the adversarial-supervised attention network until it converges. In the loop, we first inject adversarial edges into $\mathcal{G}^*$ according to Eq.(15), and the time complexity is $O(N^2)$. According to (Velickovic et al., 2018), the time complexity of a single GAT attention head is $O(Nd^{(l-1)}d^{(l)} + |\mathcal{E}|d^{(l)})$, and thus the total cost of the multi-head GAT training is $O(e_2(Nd_2^2 + |\mathcal{E}|d_2))$, where d_2 denotes the maximum dimension of features in the GAT model and e_2 denotes the number of epochs of training. Finally, the complexity of the row-wise softmax operation on H^* is $O(Nc)$, where c is the number of

classes of the dataset. As $c \ll N$, the total time complexity of ASGAT network is $O(e_2 d_2(Nd_2 + |\mathcal{E}|) + N^2)$.

Based on the analysis above, the total time complexity of DualRGNN is $O(e_1 d_1 N(d_1 + |\mathcal{E}|) + e_2 d_2(Nd_2 + |\mathcal{E}|) + N^2)$.

4. Experiments

4.1. Datasets

To evaluate the effectiveness of our DualRGNN in defending against graph adversarial attacks, we conduct extensive experiments on four widely used graph datasets, including Cora, Citeseer, Cora-ML (Sen et al., 2008), and Polblogs (Adamic and Glance, 2005). The Cora, Citeseer, and Cora-ML datasets are citation graphs, in which each node represents scientific literature, and the edges between nodes represent the citation relationship of scientific literature. The Polblogs dataset is a blog graph, in which each edge represents the link between blogs. Note that there are no node features available in the Polblogs dataset, so following the previous work (Jin et al., 2020b; Liu et al., 2021; Li et al., 2022), we set the feature matrix to be a $N \times N$ identity matrix, where N is the number of nodes. More details of these datasets are summarized in Table 1.

Table 1: Datasets statistics.

Dataset	Nodes	Edges	Features	Classes	Training / Validating / Testing
Cora	2485	5069	1433	7	247 / 249 / 1988
Citeseer	2110	3668	3703	6	210 / 211 / 1688
Cora-ML	2810	7981	2879	7	280 / 282 / 2248
Polblogs	1222	16714	/	2	121 / 123 / 978

4.2. Baselines Methods

To highlight the outstanding performance in resisting various graph adversarial attacks of our method, we compare the DualRGNN with representative and state-of-the-art graph neural networks and robust graph neural network models. More detailed descriptions of the baselines are as follows:

- GCNs (Kipf and Welling, 2017): They are the graph convolutional networks, which employed a spectral-based convolution filter designed based on the graph Laplacian operator to aggregate neighborhood features, and it is not robust for resisting graph adversarial attacks.
- GATs (Velickovic et al., 2018): They are the graph attention networks, which introduce a self-attention mechanism to specify different attention weights to different nodes when performing neighborhood aggregation.
- SGC (Wu et al., 2019a): SGC removes the nonlinear activation function and collapses weight matrices from GCNs, and also does not have a design aimed at improving robustness.

- Jaccard-GCN (Wu et al., 2019b): Jaccard-GCN adopted a Jaccard similarity to check the similarity between the node pairs of each edge and removed those edges with low similarity scores to obtain a clean graph.
- GCN-SVD (Entezari et al., 2020): GCN-SVD reconstructed the input graph with the low-rank approximation of the adjacency.
- RGCN (Zhu et al., 2019): RGCN adopted Gaussian distributions as the hidden representations of nodes in each GCNs layer and designed a variance-based attention mechanism to absorb the adversarial attacks.
- GNNGuard (Zhang and Zitnik, 2020): GNNGuard pruned edges between unrelated nodes in each GNNs layer, and simultaneously introduced a layer-wise graph memory module to stabilize training.
- Pro-GNN (Jin et al., 2020b): Pro-GNN utilized the inherent characteristics of real-world clean graphs to constrain to learn robust graph structures.
- SimP-GCN (Jin et al., 2021): SimP-GCN introduced an aggregation mechanism to maintain feature similarity, and adaptively integrates the original graph and the kNN graph.
- ElasticGNN (Liu et al., 2021): ElasticGNN introduced L1 and L2 regularization to GNNs and provided an elastic message passing scheme to enhance the local smoothness.
- STABLE (Li et al., 2022): STABLE refined the graph structure by learning representations that carry both feature and structure information and are insensitive to perturbations and simultaneously designed a robust normalization mechanism to improve GCNs.

4.3. Experimental Setting

For the baseline methods, we adopt their default parameter settings from the initial implementation. Specifically, for GNNGuard (Zhang and Zitnik, 2020), we choose GCN as the surrogate model.

In the SPGR module, we adopt a one-layer similarity-preserved graph convolution network to learn the node representations. We construct the kNN graph based on the kNN algorithm, where k is tuned from $[5, 30]$. And we employ two augmented graphs with slight adversarial perturbations for graph contrastive learning. The predefined threshold δ_1 is the Jaccard similarity threshold, which is tuned from $\{0.0, 0.01, 0.02, 0.03, 0.04, 0.05\}$. And δ_2 is cosine similarity threshold tuned from $\{0.1, 0.2, 0.3\}$. The hyper-parameter K in Top-K matrix S^K is tuned from $[2, 20]$, and the recovery portion p is fixed at 0.2.

For the architecture of the ASGAT, we adopt a two-layer graph attention network with multi-head attention. The ratio of injected adversarial edges q is tuned from $[0.02, 0.2]$ with an interval of 0.02. The hyper-parameter η is set to 100. The values of λ and μ are tuned from $[0.5, 2]$. The dropout rate is tuned from $\{0.3, 0.4, 0.5, 0.6, 0.7\}$. The number of attention heads is tuned from $[1, 8]$.

We employ Xavier algorithm (Glorot and Bengio, 2010) for the initialization of trainable parameters in DualRGNN, and adopt Adam algorithm (Kingma and Ba, 2015) for optimization, with learning rate tuned from $[0.0001, 0.1]$. The hyper-parameter weight decay is set to 0.0005. In both modules, we apply an early stopping strategy on the binary cross-entropy loss (SPGR) or both the cross-entropy loss and accuracy (ASGAT) on the validation nodes, with a patience of 100 epochs. Following the previous work (Jin et al., 2020b), the proposed DualRGNN is implemented based on the DeepRobust (Li et al., 2020) framework.

4.4. Experimental Results and Analysis

We evaluate the robustness of DualRGNN and baselines to defend against non-targeted, targeted, and random graph adversarial attacks based on the accuracy of semi-supervised graph node classification on the above four benchmark datasets. Higher classification accuracy indicates better resistance to graph adversarial attacks, reflecting higher robustness.

4.4.1. Against non-targeted graph adversarial attack

To evaluate the effectiveness of DualRGNN and baselines in resisting non-target graph adversarial attacks, Metattack (Zügner and Günnemann, 2019) is employed as the non-target graph adversarial attack method, which is an effective attack method. We set the perturbation rate of Metattack from 0% to 25% with an interval of 5%, and adopted its default parameter settings from the initial implementation.

Table 2 shows the performance of DualRGNN and baselines on four datasets under Metattack (Zügner and Günnemann, 2019) with different perturbation rates. All the reported results are averaged over 10 runs, and the top two performance is highlighted in bold and underlined. From these results, we have the following observations:

- On all of the four datasets, DualRGNN significantly outperforms all the baseline approaches in most cases, and it can maintain high semi-supervised graph node classification accuracy, especially in cases of high perturbation rates. Even compared with the recent state-of-the-art robust methods, including STABLE (Li et al., 2022) and ElasticGNN (Liu et al., 2021), DualRGNN has better results and achieves at most 1.12%, 0.67%, 0.66%, and 4.63% improvements on the Cora, Citeseer, Cora-ML, and Polblogs datasets in the perturbation rate of 25%. From the averaged performance, DualRGNN outperforms the baselines and shows its stability under different perturbation rates. These prove the high robustness of DualRGNN in resisting non-target graph adversarial attacks.

- DualRGNN can achieve satisfactory performance on clean graph data. For instance, compared with the robust methods, such as Jaccard-GCN (Wu et al., 2019b) and GNNGuard (Zhang and Zitnik, 2020), DualRGNN achieves higher accuracy in semi-supervised graph node classification on clean graph data. This demonstrates that DualRGNN will not sacrifice its performance on clean graph data in exchange for robustness against graph adversarial attacks.

- Vanilla GNN models without robust design, such as GCNs (Kipf and Welling, 2017), GATs (Velickovic et al., 2018), SGC (Wu et al., 2019a), etc., suffer severe

Table 2: Node classification performance (Accuracy±Std %) on four datasets under non-targeted attack (Metattack) with different perturbation rates. To ensure fair comparisons, we employ the same perturbed graph for each perturbation rate as the input for both our method (DualRGNN) and the baselines.

Datasets	Ptb. Rate	0%	5%	10%	15%	20%	25%	Average
Cora	GCNs (Kipf and Welling, 2017)	83.40±0.56	77.21±1.31	70.30±2.03	65.31±1.96	53.96±1.54	49.30±1.84	66.58
	GATs (Velickovic et al., 2018)	84.71±0.36	79.36±1.28	74.54±1.29	71.20±1.30	58.55±1.15	53.71±1.96	70.34
	SGC (Wu et al., 2019a)	83.80±0.10	76.40±0.16	69.08±0.35	65.66±1.56	58.32±0.11	50.94±0.26	67.37
	Jaccard-GCN (Wu et al., 2019b)	82.07±0.72	78.97±0.68	74.52±1.10	70.04±1.28	65.13±1.71	60.90±1.80	71.94
	GCN-SVD (Entezari et al., 2020)	71.62±0.45	71.01±0.54	69.43±0.67	67.30±0.76	58.94±1.13	56.20±1.47	65.75
	RGCN (Zhu et al., 2019)	83.63±0.18	76.11±0.66	72.68±0.83	68.48±0.70	57.75±0.66	53.35±0.67	68.67
	GNNGuard (Zhang and Zitnik, 2020)	78.16±0.25	77.58±0.40	75.25±0.87	74.28±1.25	74.55±0.68	71.62±1.20	75.24
	Pro-GNN (Jin et al., 2020b)	85.15±0.31	81.18±0.33	72.59±1.52	67.80±0.94	56.37±1.87	50.02±1.00	68.85
	SimP-GCN (Jin et al., 2021)	81.86±0.55	77.06±1.86	74.22±0.77	72.89±3.22	70.98±3.84	68.56±6.71	74.26
	ElasticGNN (Liu et al., 2021)	84.66±0.52	**81.63±1.29**	77.55±2.18	77.34±1.19	67.82±1.49	55.71±1.76	74.12
	STABLE (Li et al., 2022)	**85.26±0.69**	80.83±0.47	80.91±0.43	79.00±0.84	**78.38±0.43**	76.00±0.71	80.06
	DualRGNN (ours)	83.71±0.25	81.29±0.37	**81.40±1.44**	**79.17±0.27**	77.64±0.41	**77.12±0.34**	**80.12**
Citeseer	GCNs (Kipf and Welling, 2017)	72.00±0.38	70.80±0.94	67.17±1.06	64.34±0.96	55.49±1.86	55.41±2.36	64.20
	GATs (Velickovic et al., 2018)	73.31±0.70	72.30±1.13	70.04±0.82	68.32±1.39	60.26±1.14	61.75±1.10	67.66
	SGC (Wu et al., 2019a)	73.85±0.21	72.11±0.29	66.10±0.49	64.31±1.62	54.35±0.68	50.70±1.44	63.57
	Jaccard-GCN (Wu et al., 2019b)	72.20±0.57	70.36±1.10	68.22±1.07	66.13±0.96	60.15±0.91	59.22±1.77	66.05
	GCN-SVD (Entezari et al., 2020)	67.63±1.14	68.61±0.61	69.46±0.81	67.75±1.20	68.40±0.57	64.69±0.95	67.76
	RGCN (Zhu et al., 2019)	72.81±0.46	71.71±0.56	69.13±0.32	65.34±0.81	56.39±0.88	58.29±1.21	65.61
	GNNGuard (Zhang and Zitnik, 2020)	70.57±0.98	69.32±1.47	70.58±0.98	69.40±0.75	69.28±0.91	68.46±1.08	69.60
	Pro-GNN (Jin et al., 2020b)	73.24±0.68	71.55±0.54	69.64±0.81	65.97±0.73	55.76±1.41	56.36±2.49	65.42
	SimP-GCN (Jin et al., 2021)	74.30±0.54	73.90±0.72	72.04±1.63	71.65±1.23	67.06±4.09	70.56±2.25	71.58
	ElasticGNN (Liu et al., 2021)	73.64±0.52	72.92±0.34	72.23±0.49	70.91±0.76	57.99±1.84	59.77±4.05	67.91
	STABLE (Li et al., 2022)	74.03±0.62	73.93±0.42	72.58±0.63	73.32±1.00	72.80±0.58	72.62±0.70	73.21
	DualRGNN (ours)	**74.49±0.33**	**75.74±0.55**	**73.59±0.41**	**74.97±0.66**	**73.05±0.25**	**73.29±0.55**	**74.19**
Cora-ML	GCNs (Kipf and Welling, 2017)	85.72±0.19	80.32±0.25	74.27±0.37	53.98±0.80	45.41±0.71	49.20±0.51	64.82
	GATs (Velickovic et al., 2018)	85.48±0.28	81.18±0.95	76.47±0.61	57.45±0.97	42.29±2.14	45.38±3.64	64.71
	SGC (Wu et al., 2019a)	82.43±0.00	70.93±0.02	54.76±0.46	47.84±0.25	38.15±0.25	44.98±0.31	56.62
	Jaccard-GCN (Wu et al., 2019b)	84.74±0.23	80.20±0.46	75.17±0.41	57.14±1.16	46.76±1.04	49.05±0.32	65.51
	GCN-SVD (Entezari et al., 2020)	80.93±0.33	80.08±0.38	80.54±0.30	74.08±0.98	47.42±0.92	56.41±0.75	69.91
	RGCN (Zhu et al., 2019)	86.31±0.27	81.30±0.37	74.77±0.47	55.03±0.50	47.32±0.35	50.58±0.14	65.88
	GNNGuard (Zhang and Zitnik, 2020)	76.83±0.57	76.25±0.54	76.44±0.50	75.82±0.60	73.53±0.45	73.20±0.83	75.34
	Pro-GNN (Jin et al., 2020b)	85.32±0.44	83.75±0.49	81.88±0.42	53.88±0.22	48.67±0.34	50.33±0.64	67.30
	SimP-GCN (Jin et al., 2021)	85.34±0.37	83.53±0.51	80.64±0.54	76.72±5.97	69.4±11.70	69.16±12.42	77.46
	ElasticGNN (Liu et al., 2021)	**86.63±0.45**	83.77±0.94	81.03±0.92	71.50±1.70	52.69±0.66	52.79±0.39	71.40
	STABLE (Li et al., 2022)	85.63±0.29	83.64±0.39	82.51±0.30	81.11±0.43	80.86±0.43	80.89±0.28	82.44
	DualRGNN (ours)	86.18±0.21	**84.08±0.32**	**84.24±0.34**	**81.98±0.46**	**82.05±0.27**	**81.55±0.30**	**83.35**
Polblogs	GCNs (Kipf and Welling, 2017)	95.71±0.22	72.51±0.63	72.49±0.79	68.16±0.84	58.85±3.56	56.64±2.64	70.73
	GATs (Velickovic et al., 2018)	95.32±0.42	76.64±0.75	72.78±0.77	59.07±6.68	51.72±0.37	49.58±4.39	67.52
	SGC (Wu et al., 2019a)	94.45±0.23	74.56±0.46	70.52±0.15	55.96±1.95	51.88±0.15	51.82±0.32	66.53
	Jaccard-GCN (Wu et al., 2019b)	/	/	/	/	/	/	/
	GCN-SVD (Entezari et al., 2020)	94.52±0.23	77.77±0.97	69.50±3.01	64.22±0.81	54.35±1.78	54.57±1.21	69.16
	RGCN (Zhu et al., 2019)	95.31±0.18	71.69±0.47	71.18±0.76	67.19±1.26	62.04±2.95	58.08±0.71	70.92
	GNNGuard (Zhang and Zitnik, 2020)	/	/	/	/	/	/	/
	Pro-GNN (Jin et al., 2020b)	95.60±0.23	81.34±5.85	72.17±2.08	67.87±1.94	54.89±3.86	52.17±1.59	70.67
	SimP-GCN (Jin et al., 2021)	94.89±1.13	71.83±0.99	72.12±0.72	69.85±1.20	58.52±2.96	57.45±2.51	70.78
	ElasticGNN (Liu et al., 2021)	**95.77±0.31**	74.52±1.17	75.82±0.75	73.96±0.60	71.87±1.31	52.02±0.06	73.99
	STABLE (Li et al., 2022)	95.64±0.19	81.67±1.08	79.07±0.71	72.58±1.31	70.42±0.57	66.86±0.89	77.71
	DuanRGNN (ours)	95.52±0.14	**83.22±2.57**	**79.61±0.51**	**75.45±1.36**	**72.16±1.26**	**71.49±1.36**	**79.58**

performance degradation on perturbed graph data, which indicates the weak robustness of traditional GNNs in resisting graph adversarial attacks. The robust GNN methods can still achieve satisfactory semi-supervised graph node classification performance on perturbed graphs even with high perturbation rates.

To further show the robustness of our model against diverse attacks, we expand our experiments by evaluating our model under DICE (Waniek et al., 2018) attack, also known as 'delete internally, connect externally'. In each perturbation, DICE randomly decides whether to insert or remove an edge. Notably, edges are only removed between nodes of the same class, and only inserted between nodes of different classes. We choose the Cora-ML dataset for evaluation in this section, vary the perturbation rate of the DICE attack from 0% to 25% with an interval of 5%, and adopt its default parameter settings from the initial implementation. Table 3 shows the performance results of DualRGNN and baselines on the Cora-ML dataset under the DICE attack with different perturbation rates. All the reported results are averaged over 10 runs, and the top two performances are respectively highlighted in bold and underlined. From these results, we can observe that DualRGNN can achieve comparable performance in most cases under the DICE attack. While the performance of DualRGNN shows a slight decrease compared to GNNGuard (Zhang and Zitnik, 2020) when the perturbation rate is high, it excels as the top performer on average.

Table 3: Node classification performance (Accuracy±Std %) on Cora-ML dataset under non-targeted attack (DICE) with different perturbation rates. To ensure fair comparisons, we employ the same perturbed graph for each perturbation rate as the input for both our method (DualRGNN) and the baselines.

Ptb. Rate	0%	5%	10%	15%	20%	25%	Average
GCNs (Kipf and Welling, 2017)	85.72±0.19	84.26±0.28	83.28±0.36	81.97±0.27	80.28±0.50	80.18±0.23	82.62
GATs (Velickovic et al., 2018)	85.48±0.28	83.82±0.61	82.60±0.38	81.11±0.46	79.18±0.72	77.99±0.36	81.70
SGC (Wu et al., 2019a)	82.43±0.00	78.95±0.02	75.04±0.00	73.79±0.02	71.17±0.07	69.91±0.02	75.22
Jaccard-GCN (Wu et al., 2019b)	84.74±0.23	83.28±0.14	82.02±0.25	81.33±0.36	79.74±0.36	78.84±0.15	81.66
GCN-SVD (Entezari et al., 2020)	80.93±0.33	80.77±0.25	79.31±0.39	77.91±0.22	77.31±0.42	76.32±0.28	78.77
RGCN (Zhu et al., 2019)	86.31±0.27	84.61±0.20	83.55±0.34	82.24±0.23	80.74±0.35	79.92±0.46	82.90
GNNGuard (Zhang and Zitnik, 2020)	76.83±0.57	82.67±0.17	82.56±0.30	**82.75±0.19**	**82.75±0.18**	**82.74±0.24**	81.72
Pro-GNN (Jin et al., 2020b)	85.32±0.44	79.14±0.70	78.32±0.62	76.20±1.71	74.91±1.60	73.51±1.83	77.90
SimP-GCN (Jin et al., 2021)	85.34±0.37	83.73±0.41	83.00±0.49	81.88±0.45	80.97±0.48	81.06±0.35	82.66
ElasticGNN (Liu et al., 2021)	**86.63±0.45**	73.35±1.78	75.76±0.99	76.69±1.85	74.61±2.03	76.54±1.46	77.26
STABLE (Li et al., 2022)	85.63±0.29	81.98±1.13	80.86±0.92	80.32±0.72	78.96±0.69	78.23±0.6	81.00
DualRGNN (ours)	86.18±0.21	**84.65±0.77**	**83.89±0.48**	82.51±0.95	81.94±1.02	81.95±0.69	**83.52**

4.4.2. Against targeted graph adversarial attack

To evaluate the effectiveness of DualRGNN in defending against target graph adversarial attacks, we employ Nettack (Zügner et al., 2018) as the target graph adversarial attack method, and we follow the default parameter settings in the authors' original implementation. Following (Zhu et al., 2019; Jin et al., 2020b), we set the number of perturbations made on every targeted node from 0 to 5 with a step size of 1. The nodes in the test set with a degree larger than 10 are treated as target nodes.

Table 4 reports the semi-supervised graph node classification accuracy of all baseline methods on four datasets under Nettack (Zügner et al., 2018) attack with different numbers of perturbations. Due to the node features being unavailable on the Polblogs dataset, it is no means of evaluating the baselines methods Jaccard-GCN (Wu et al.,

2019b) and GNNGuard (Zhang and Zitnik, 2020) on this dataset. For our DualRGNN, a random kNN graph is constructed on the Polblogs dataset as one of the inputs for the SPGR module. Similarly, all the reported results are averaged over 10 runs, and the top two performance is highlighted in bold and underlined. From these results, we can make a few observations as follows:

- On the Cora and Citeseer datasets, our DualRGNN approach can effectively defend against target graph adversarial attacks, and outperform all the baseline methods in most cases. Compared with STABLE (Li et al., 2022) and ElasticGNN (Liu et al., 2021) methods, DualRGNN can achieve higher semi-supervised graph node classification accuracy, especially in situations where the target node has a higher number of perturbations. For example, on the Cora and Citeseer datasets at 5 perturbations per targeted node, DualRGNN outperforms them by 2.17% and 2.69%, respectively. It can also be noted that DualRGNN exhibits superior performance on average. These show the high robustness of DualRGNN in defending against target graph adversarial attacks.

- On the Cora-ML dataset, DualRGNN did not outperform some baseline methods like GNNGuard (Zhang and Zitnik, 2020) and ElasticGNN (Liu et al., 2021) on perturbed graphs. Since the Cora-ML dataset is larger than the other three datasets and DualRGNN outperforms the baseline methods under Metattack on the same dataset (shown in Table 2), DualRGNN faces challenge in some scenarios, particularly on larger datasets under targeted attacks. Despite this limitation, the overall performance of DualRGNN remains comparable to most of the baseline methods.

- On the Polblogs dataset, DualRGNN has also achieved high performance. Even if the node features are unavailable on this dataset, DualRGNN cannot model the original node similarity relationship based on the kNN graph. However, compared to most existing graph neural networks and robust graph neural network models, DualRGNN still achieves acceptable semi-supervised graph node classification accuracy. This is because on the Polblogs dataset, constructing a random kNN graph for DualRGNN can improve the model's generalization ability to a certain extent, reduce its dependence on input graphs, and thus weaken the impact of attacked graph data on model performance.

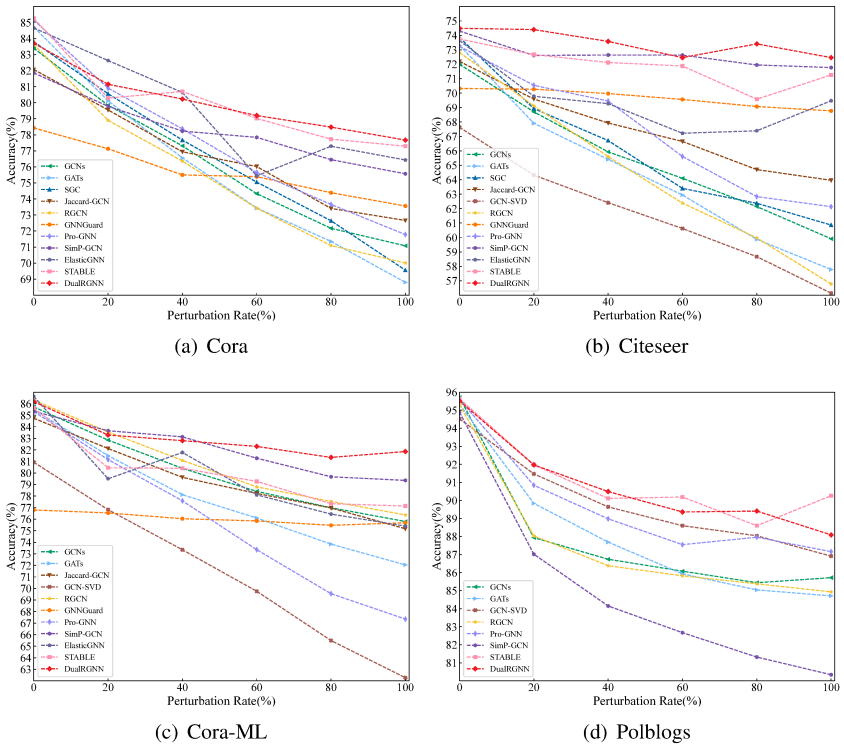
4.4.3. Against random graph adversarial attack

We evaluate how DualRGNN behaves under different ratios of random noises, and we vary the perturbation rate from 0% to 100% with a step size of 20%. We conduct experiments on four datasets, and the result is reported in Fig. 3. From the result, we make the following observations:

- Although DualRGNN cannot fully play its advantages in a clean graph, it can still achieve high semi-supervised graph node classification accuracy on perturbed graph data under high perturbation rates, significantly outperforming the existing methods. For instance, on the Cora, Citeseer, Cora-ML, and Polblogs

Table 4: Node classification performance (Accuracy±Std %) on four datasets under targeted attack (Nettack) with different number of perturbations. To ensure fair comparisons, we employ the same perturbed graph for each number of perturbations as the input for both our method (DualRGNN) and the baselines.

Datasets	No. of Ptb.	0	1	2	3	4	5	Average
Cora	GCNs (Kipf and Welling, 2017)	79.52±2.72	75.06±1.97	70.72±1.14	67.71±2.11	61.45±1.50	55.18±1.37	68.27
	GATs (Velickovic et al., 2018)	81.57±2.54	77.59±3.86	70.00±4.49	67.35±2.98	57.23±4.44	53.98±5.23	67.95
	SGC (Wu et al., 2019a)	83.01±0.68	74.58±0.89	71.08±0.57	67.23±0.51	63.61±0.76	55.54±1.33	69.18
	Jaccard-GCN (Wu et al., 2019b)	78.92±2.49	75.54±3.46	70.24±2.54	66.27±1.88	60.00±1.59	61.08±2.84	68.67
	GCN-SVD (Entezari et al., 2020)	71.62±0.45	65.42±3.59	63.49±2.61	64.58±1.30	54.46±1.95	60.24±1.70	63.30
	RGCN (Zhu et al., 2019)	81.20±1.52	78.31±1.80	71.33±1.24	67.59±1.84	60.84±1.73	55.78±1.80	69.17
	GNNGuard (Zhang and Zitnik, 2020)	76.39±2.55	72.41±2.44	73.01±1.90	75.06±2.61	69.88±3.16	69.04±2.73	72.63
	Pro-GNN (Jin et al., 2020b)	**84.94±1.90**	81.93±1.14	73.86±0.81	71.33±0.95	66.51±2.40	60.00±1.59	73.09
	SimP-GCN (Jin et al., 2021)	80.72±1.80	77.59±1.16	74.10±2.85	71.20±2.92	65.06±3.11	57.83±3.21	71.08
	ElasticGNN (Liu et al., 2021)	83.61±2.55	80.12±2.43	77.95±1.61	75.42±1.90	71.45±3.27	66.99±2.06	75.92
	STABLE (Li et al., 2022)	84.10±1.37	81.93±2.27	**80.48±2.18**	78.43±1.92	72.41±1.33	72.05±1.87	78.23
	DualRGNN (ours)	84.34±1.80	**82.29±1.51**	79.64±1.20	**78.55±1.11**	**73.97±1.62**	**74.22±2.06**	**78.83**
Citeseer	GCNs (Kipf and Welling, 2017)	80.16±1.87	78.89±1.84	76.19±4.03	65.56±3.09	59.21±3.18	50.95±6.88	68.49
	GATs (Velickovic et al., 2018)	81.75±0.84	77.30±3.51	64.76±12.11	51.43±6.53	53.97±7.14	43.97±8.76	62.20
	SGC (Wu et al., 2019a)	80.95±0.00	77.78±0.00	76.19±0.00	74.60±0.00	66.67±0.00	61.90±0.00	73.02
	Jaccard-GCN (Wu et al., 2019b)	80.79±1.39	78.25±1.31	77.46±1.95	73.97±3.19	65.40±4.41	60.32±3.82	72.70
	GCN-SVD (Entezari et al., 2020)	80.32±1.71	75.71±2.90	68.41±5.57	61.11±12.19	57.30±4.39	54.29±11.04	66.19
	RGCN (Zhu et al., 2019)	80.95±0.00	79.37±0.75	77.78±1.50	60.95±1.71	57.94±1.12	49.37±1.39	67.73
	GNNGuard (Zhang and Zitnik, 2020)	82.06±3.09	79.68±2.34	77.46±2.88	80.32±4.50	77.78±3.51	75.56±8.10	78.81
	Pro-GNN (Jin et al., 2020b)	82.22±0.67	81.11±1.39	79.05±0.71	76.83±2.66	68.57±2.07	61.27±9.02	74.84
	SimP-GCN (Jin et al., 2021)	80.95±0.75	79.68±1.25	78.89±2.90	74.76±4.58	72.38±7.45	71.43±7.37	76.35
	ElasticGNN (Liu et al., 2021)	81.59±0.82	80.95±0.00	80.63±0.67	79.05±2.22	78.25±2.12	73.65±3.10	79.02
	STABLE (Li et al., 2022)	82.06±0.77	81.59±0.82	80.79±1.90	80.63±1.00	75.87±3.95	68.41±5.37	78.23
	DualRGNN (ours)	**82.70±0.90**	**81.75±1.12**	**81.27±1.25**	**80.64±1.00**	**81.27±1.80**	**78.25±2.60**	**80.98**
Cora-ML	GCNs (Kipf and Welling, 2017)	86.21±0.20	82.08±0.57	75.72±1.03	70.26±0.47	64.98±1.38	58.36±2.70	72.94
	GATs (Velickovic et al., 2018)	86.77±0.71	80.86±3.10	73.27±5.62	71.12±3.46	63.49±4.60	62.23±2.94	72.96
	SGC (Wu et al., 2019a)	87.36±0.00	84.01±0.00	78.81±0.00	72.86±0.00	67.29±0.00	60.59±0.00	75.15
	Jaccard-GCN (Wu et al., 2019b)	85.54±0.59	81.38±0.31	74.83±0.62	70.74±1.13	63.46±1.09	59.41±1.37	72.56
	GCN-SVD (Entezari et al., 2020)	84.28±0.58	81.71±0.52	76.36±0.88	70.07±0.42	68.40±1.00	65.39±1.70	74.37
	RGCN (Zhu et al., 2019)	87.21±0.45	82.83±0.46	76.47±1.03	69.85±1.31	61.04±1.70	57.03±1.54	72.41
	GNNGuard (Zhang and Zitnik, 2020)	83.53±0.41	83.64±0.58	83.90±0.41	**82.75±0.67**	**82.94±0.61**	**83.05±0.90**	**83.30**
	Pro-GNN (Jin et al., 2020b)	79.64±0.63	79.59±1.49	78.66±1.59	74.57±1.71	71.74±1.84	67.55±2.99	75.29
	SimP-GCN (Jin et al., 2021)	84.61±1.21	81.12±0.94	76.99±1.02	72.27±0.83	66.91±1.65	65.43±1.78	74.56
	ElasticGNN (Liu et al., 2021)	87.66±0.89	**86.21±0.87**	**84.16±0.40**	81.71±0.96	78.03±1.28	74.83±1.23	82.10
	STABLE (Li et al., 2022)	86.93±0.89	84.47±0.93	82.19±2.15	78.65±1.76	75.64±0.88	70.97±1.81	79.81
	DualRGNN (ours)	**87.73±0.89**	85.47±0.81	83.12±0.77	80.48±1.15	80.37±1.60	79.15±0.92	82.72
Polblogs	GCNs (Kipf and Welling, 2017)	97.17±0.15	96.94±0.18	95.78±0.23	95.44±0.32	94.20±0.21	93.11±0.31	95.44
	GATs (Velickovic et al., 2018)	97.35±0.33	97.41±0.26	96.26±0.32	95.52±0.81	95.00±0.74	91.59±1.27	95.52
	SGC (Wu et al., 2019a)	96.85±0.00	96.67±0.00	96.56±0.10	95.96±0.15	94.85±0.29	89.72±1.65	95.10
	Jaccard-GCN (Wu et al., 2019b)	/	/	/	/	/	/	/
	GCN-SVD (Entezari et al., 2020)	**97.50±0.18**	97.89±0.26	97.65±0.21	**97.28±0.15**	**96.56±0.13**	95.41±0.27	97.05
	RGCN (Zhu et al., 2019)	97.15±0.22	97.02±0.06	95.63±0.23	95.28±0.24	94.28±0.16	92.72±0.20	95.35
	GNNGuard (Zhang and Zitnik, 2020)	/	/	/	/	/	/	/
	Pro-GNN (Jin et al., 2020b)	97.44±0.08	97.74±0.27	97.33±0.21	96.93±0.21	96.19±0.17	95.30±0.93	96.82
	SimP-GCN (Jin et al., 2021)	91.04±0.52	96.50±0.31	94.67±0.38	94.22±0.26	93.15±0.55	91.04±0.52	93.44
	ElasticGNN (Liu et al., 2021)	97.44±0.21	96.00±2.07	94.22±2.74	95.33±0.83	95.56±0.56	91.26±7.89	94.97
	STABLE (Li et al., 2022)	97.37±0.19	97.91±0.12	97.59±0.23	91.76±1.52	87.24±1.08	86.17±1.71	93.01
	DualRGNN (ours)	97.43±0.24	**97.98±0.14**	**97.72±0.26**	97.05±0.25	96.35±0.15	**96.30±0.30**	**97.14**



(a) Cora (b) Citeseer

(c) Cora-ML (d) Polblogs

Figure 3: Node classification performance (Accuracy %) on four datasets under random attack with different perturbation rates. To ensure fair comparisons, we employ the same perturbed graph for each perturbation rate as the input for both our method (DualRGNN) and the baselines.

datasets with 80% perturbation rate, compared with the state-of-the-art methods, DualRGNN achieved improvements of 0.75%, 1.46%, 1.68%, and 0.81%, respectively.

- With the increases of perturbation rates, the classification performance of vanilla GNN models, including (Kipf and Welling, 2017), GATs (Velickovic et al., 2018) and SGC (Wu et al., 2019a), degrades quickly, indicating poor robustness against random graph adversarial attacks. The pre-processing methods have a weak ability to resist random graph adversarial attacks, especially in situations when the perturbed graph is under high perturbation rates. Specifically, under the 100% perturbation rate on the Cora-ML dataset, GCN-SVD (Entezari et al., 2020) is reduced by nearly 20% compared to DualRGNN. This is because the pre-processing methods only preprocess the input graph once, which cannot effectively recover the graph structure under high perturbation rates. In addition, GCN-SVD (Entezari et al., 2020) is mainly designed to resist target graph adversarial attacks and is poor against random attacks.

4.5. Ablation Study

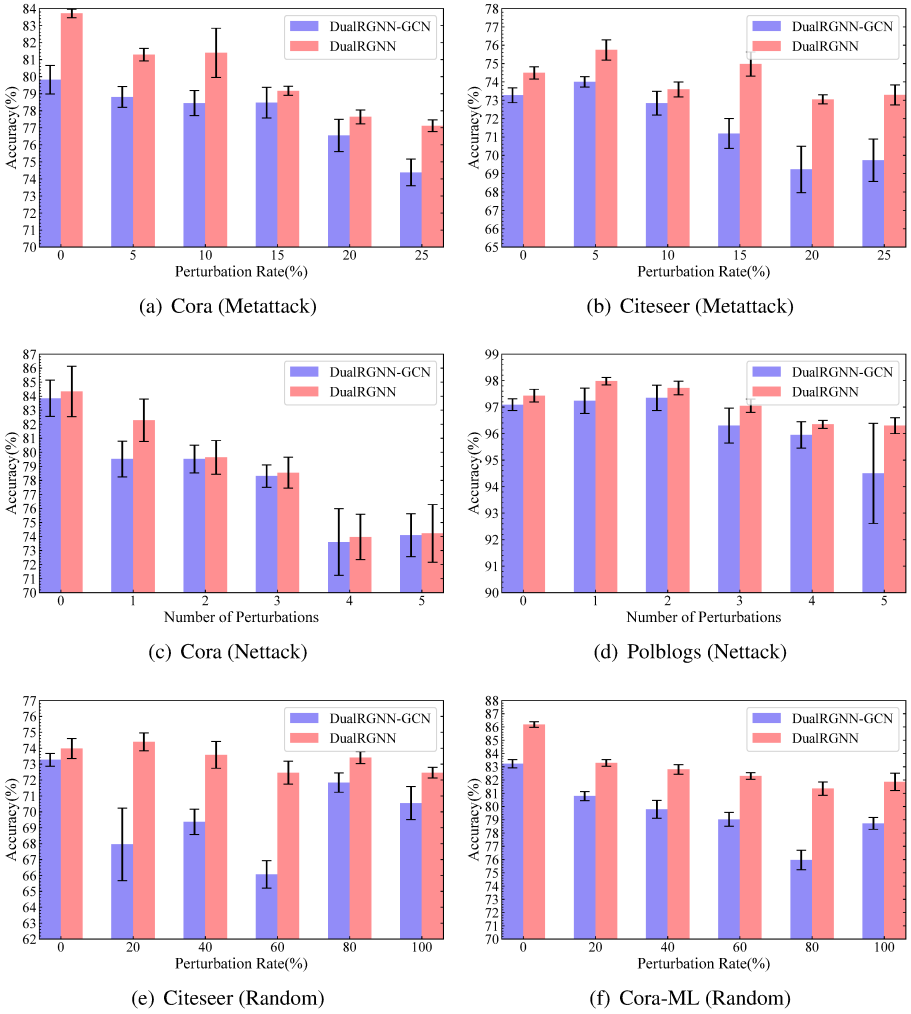
4.5.1. Effectiveness of node similarity preserving graph refinining

In this work, we propose refining the graph structure based on the learned node representations that preserve the original nodes' similarity, aiming to mitigate the poisoning of graph adversarial attacks on graph data. In order to verify whether learning node representations preserving the original node similarity relationship for graph structure refinement is beneficial for improving the robustness of DualRGNN against graph adversarial attacks, we conduct an ablation study. We replace the similarity-preserved graph convolution with vanilla GCNs and denoted it as *DualRGNN-GCN*. This implies that the *DualRGNN-GCN* employed traditional GCNs for node representation learning, and the learned node representations cannot preserve the node similarity relationships in the original feature space. The ablation experiments are conducted on the Cora, Citeseer, Cora-ML, and Polblogs datasets to defend against non-targeted, targeted, and random attacks. The comparison results between *DualRGNN-GCN* and DualRGNN are shown in Fig. 4.

From these results, we can observe that refining the graph structure based on the learned node representations that preserve the original nodes' similarity can effectively improve the ability of DualRGNN in resisting various graph adversarial attacks, especially the non-target and random attacks. The proposed DualRGNN model demonstrates robust and stable performance, especially on the datasets like Citeseer with random noise and Metattack. In these cases, the accuracy of DualRGNN remains consistently high in different perturbation rates. For the Cora dataset under Metattack and the Cora-ML dataset under random attack, although both DualRGNN and vanilla GCNs experience a decrease in performance as the perturbation rate increases, DualRGNN performs better vanilla GCNs. When assessing model performance on the Cora and Polblogs datasets under Nettack, DualRGNN performs comparably with DualRGNN-GCN. In comparison to vanilla GCNs, DualRGNN exhibits improved robustness, a result of its effective utilization of similarity information for refining graph structures and obtaining more reliable node representations.

4.5.2. Effectiveness of adversarial-supervised graph attention network

We propose the ASGAT network, treating adversarial edges as supervisions signals and developing an adversarial-supervised attention mechanism to recognize adversarial edges suppressing the propagation of adversarial perturbation. To verify whether the proposed ASGAT is beneficial for improving the robustness of DualRGNN against graph adversarial attacks, another ablation study has been conducted. Firstly, we remove the adversarial-supervised attention mechanism from DualRGNN and adopt the traditional graph attention mechanism for graph representation learning, naming it as *DualRGNN-GAT*. This means that the *DualRGNN-GAT* lacks supervision of adversarial information, and it lacks the ability to recognize adversarial edges. Secondly, we remove the graph contrastive learning framework from DualRGNN, adopting adversarial-supervised attention mechanism but without optimizing based on graph contrastive learning, and this variant model is denoted as *DualRGNN-AGAT*. That is, the *DualRGNN-AGAT* replaces the traditional attention mechanism in *DualRGNN-GAT* with an adversarial-supervised attention mechanism, and it has a certain recognition ability for adversarial edges. The ablation experiments are conducted



(a) Cora (Metattack) (b) Citeseer (Metattack)

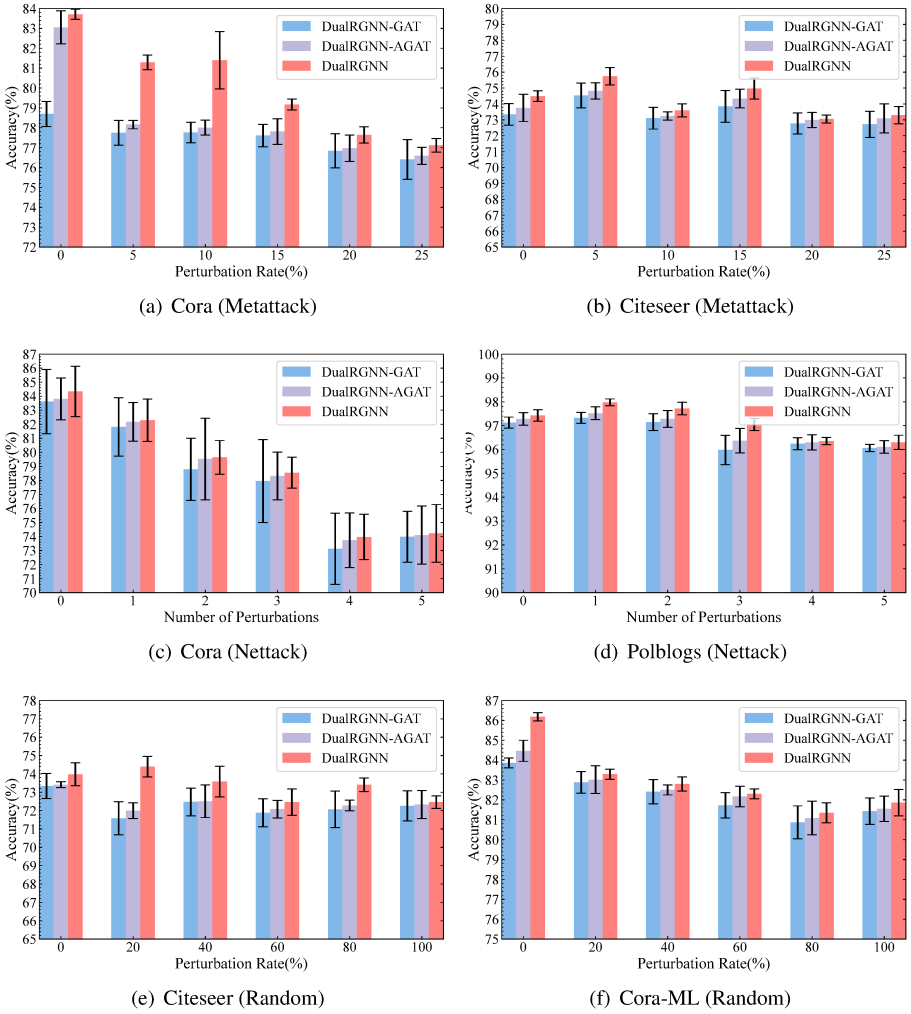
(c) Cora (Nettack) (d) Polblogs (Nettack)

(e) Citeseer (Random) (f) Cora-ML (Random)

Figure 4: Effectiveness of node similarity preserving graph refining. The average accuracy of DualRGNN-GCN and DualRGNN on Cora, Citeseer, Cora-ML, and Polblogs datasets under different graph adversarial attacks.

on all the above four benchmark datasets to resist non-target, target, and random attacks, and the comparison results are reported in Fig. 5. From these results, we can make a few observations as follows:

- DualRGNN can achieve higher classification accuracy than *DualRGNN-GAT*, and has a lower standard deviation in most cases. This indicates that introducing the ASGAT can effectively improve the performance of DualRGNN, which proves the high robustness of the proposed ASGAT in resisting various graph adversarial attacks.



(a) Cora (Metattack) (b) Citeseer (Metattack)

(c) Cora (Nettack) (d) Polblogs (Nettack)

(e) Citeseer (Random) (f) Cora-ML (Random)

Figure 5: Effectiveness of adversarial-supervised graph attention network. The average accuracy of DualRGNN-GAT, DualRGNN-AGAT, and DualRGNN on Cora, Citeseer, Cora-ML, and Polblogs datasets under different graph adversarial attacks.

- *DualRGNN-AGAT* can achieve higher predictive accuracy than *DualRGNN-GAT*, indicating that employing the adversarial-supervised attention mechanism to treat adversarial edges information as supervised signals can effectively improve DualRGNN's ability against various graph adversarial attacks, showing the effectiveness of the designed adversarial-supervised attention mechanism.

- DualRGNN can achieve higher semi-supervised graph node classification accuracy than *DualRGNN-AGAT*, which indicates that the integrated graph contrastive learning can better leverage the advantages of the adversarial-supervised

attention mechanism. This also proves the effectiveness of ASGAT in resisting graph adversarial attacks.

4.6. Parameter Analysis

4.6.1. Effect of q

We inject adversarial edges that are treated as supervised signals by randomly recovering a small portion of removed edges, and q is the hyper-parameter that controls the recovery ratio. To verify how different values of q influence the performance of DualRGNN, we conduct a parameter analysis experiment. We set the values of q from 0.02 to 0.20, and the experiment result is shown in Fig. 6. From this result, we can observe that choosing an appropriate value for q can fully leverage the performance of DualRGNN and increase its classification accuracy, which is in line with our expectations of treating adversarial edges as supervised signals. For the most part, the optimal value of q is distributed in the range of [0.06, 0.14].

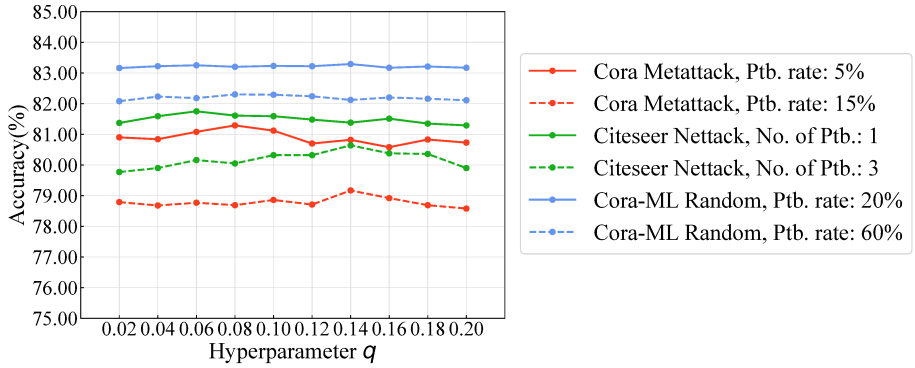


Figure 6: Results of parameter analysis: the parameter sensitivity with respect to q on Cora, Citeseer, and Cora-ML datasets under different graph adversarial attacks. Each accuracy presented is the mean of 10 independent runs.

4.6.2. Effect of λ and μ

The proposed ASGAT obtains its objective function by linearly combining the adversarial-supervised attention loss, contrastive learning loss, and cross-entropy loss, which is shown in Eq.(24), and λ and μ are the trade-off parameter. We conduct a parameter analysis experiment to verify how different values of λ and μ influence the performance of DualRGNN. For presentation simplicity, we only show results for λ and μ ranging from 0.5 to 1.8 in Figs. 7 and 8. Observing Fig. 7, it is evident that the model is insensitive to the value of the hyperparameter λ under random attack. However, under Metattack and Nettack with a higher perturbation rate, choosing an appropriate value for λ can enhance the classification accuracy of DualRGNN. Generally, optimal values for λ fall within the range of $[1.2, 1.4]$. Similar trends are observed in Fig. 8. Choosing an appropriate value for μ can also effectively improve the semi-supervised graph node classification accuracy of DualRGNN, with optimal values falling within the range of $[1.3, 1.5]$.

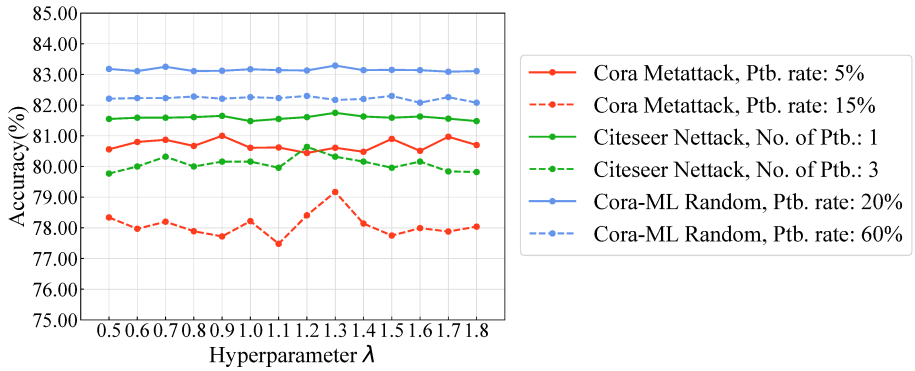


Figure 7: Results of parameter analysis: the parameter sensitivity with respect to λ on Cora, Citeseer and Cora-ML datasets under different graph adversarial attacks. Each accuracy presented is the mean of 10 independent runs.

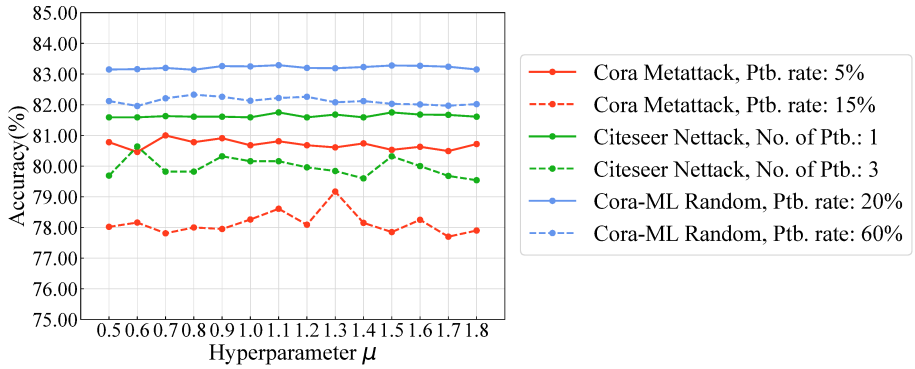


Figure 8: Results of parameter analysis: the parameter sensitivity with respect to μ on Cora, Citeseer and Cora-ML datasets under different graph adversarial attacks. Each accuracy presented is the mean of 10 independent runs.

4.7. Efficiency and Scalability Analysis

To assess the efficiency and scalability of DualRGNN, we measured the elapsed time for training the model on the Cora-ML dataset. Since Pro-GNN (Jin et al., 2020b), SimP-GCN (Jin et al., 2021), Elastic GNN (Liu et al., 2021), and STABLE (Li et al., 2022) performs well, we chose these models as the baselines in this part. Table 5 presents the elapsed time for training DualRGNN and the four baseline models under Metattack with a fixed perturbation rate of 0.1. All reported results are averaged over 10 independent runs.

From Table 5, it is evident that DualRGNN requires a longer training time compared to most baseline methods, although it is considerably faster than Pro-GNN (Jin et al., 2020b). The longer training time can be attributed to the complexity of DualRGNN, which consists of two main components and more trainable parameters compared to baseline methods that employ a single framework. For a given input graph,

Table 5: Training times (Mean±Std seconds) on Cora-ML dataset under Metattack.

Model	Training Time
ProGNN (Jin et al., 2020b)	1553.70±3.07
SimP-GCN (Jin et al., 2021)	13.74±0.65
Elastic GNN (Liu et al., 2021)	17.41±0.54
STABLE (Li et al., 2022)	15.43±0.98
DualRGNN (1 attention head)	68.46±12.51
DualRGNN (4 attention heads)	82.42±12.05
DualRGNN (8 attention heads)	91.55±14.40

the similarity-preserved graph convolution in the SPGR module needs to be trained for 300 to 600 epochs to minimize the binary cross-entropy loss $\mathcal{L}_C$, taking approximately 50 to 60 seconds. The longer training time is a result of the higher number of trainable parameters in the SPGR module, making a single epoch cost 1 to 2 seconds. After refining the graph, training the ASGAT network for 500 epochs takes 10 to 30 seconds, with the duration depending on the number of attention heads. Despite the larger time complexity of DualRGNN, the modest increase in training time is deemed acceptable, considering the improvement it brings to the performance of robust GNN models.

5. Conclusion

This paper presents a novel approach called Dual Robust Graph Neural Network (DualRGNN) that effectively defends against various graph adversarial attacks. DualRGNN comprises two key modules: SPGR and ASGAT. The SPGR module can refine the graph structure by utilizing learned node representations that preserve the similarity relationships among the original nodes. This refinement process aims to weaken the impact of graph adversarial attacks on the graph data. The ASGAT network employs an adversarial-supervised attention mechanism, where adversarial edges are treated as supervised signals. This mechanism enhances the model's ability to identify and handle adversarial edges effectively. By integrating these two modules, DualRGNN can achieve higher robustness and effectiveness in countering graph adversarial attacks. Extensive experiments were conducted on four benchmark datasets, demonstrating the effectiveness of DualRGNN against non-targeted, targeted, and random graph adversarial attacks. Additionally, an extensive ablation study is performed to validate the efficacy of each component in DualRGNN.

While our approach can achieve impressive performance under adversarial attacks, there are still some limitations to be revolved in the future. Specifically, DualRGNN involves a higher number of parameters compared to existing methods, resulting in increased time and memory consumption during the training phase. Additionally, it can be noted that DualRGNN is currently limited to node-level representation learning, and is not applicable to graph-level downstream tasks like graph classification. Addressing these challenges will be a focus for future enhancements.

For future work, we plan to extend the application of DualRGNN to defend against other types of graph adversarial attacks. Furthermore, we will explore the incorporation of hypergraphs to capture and model higher-order correlations among the data. This

approach has the potential to mitigate the impact of graph adversarial attacks on graph data to a certain extent.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant 62276101 and the National Key R&D Program of China under Grant 2019YFC1510400.

References

Adamic, L. A. and Glance, N. (2005). The political blogosphere and the 2004 u.s. election: Divided they blog. New York, NY, USA. Association for Computing Machinery.

Chen, Z., Wei, X., Wang, P., and Guo, Y. (2019). Multi-label image recognition with graph convolutional networks. In *2019 CVF/IEEE Conference on Computer Vision and Pattern Recognition*, pages 5177–5186. CVF/IEEE.

Cinà, A. E., Grosse, K., Demontis, A., Vascon, S., Zellinger, W., Moser, B. A., Oprea, A., Biggio, B., Pelillo, M., and Roli, F. (2023). Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Comput. Surv.*, 55(13s).

Dai, E., Jin, W., Liu, H., and Wang, S. (2022a). Towards robust graph neural networks for noisy graphs with sparse labels. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM '22, page 181–191, New York, NY, USA. Association for Computing Machinery.

Dai, G., Wang, X., Zou, X., Liu, C., and Cen, S. (2022b). Mrgat: Multi-relational graph attention network for knowledge graph completion. *Neural Networks*, 154:234–245.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. (2018). Adversarial attack on graph structured data. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1123–1132. PMLR.

Ding, M., Tang, J., and Zhang, J. (2018). Semi-supervised learning on graphs with generative adversarial nets. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 913–922. ACM.

Dornaika, F., Bi, J., and Zhang, C. (2023). A unified deep semi-supervised graph learning scheme based on nodes re-weighting and manifold regularization. *Neural Networks*, 158:188–196.

Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., and Papalexakis, E. E. (2020). All you need is low (rank): Defending against adversarial attacks on graphs. In *Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining*, pages 169–177.

Feng, F., He, X., Tang, J., and Chua, T. (2021). Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Trans. Knowl. Data Eng.*, 33(6):2493–2504.

Finkelshtein, B., Baskin, C., Zheltonozhskii, E., and Alon, U. (2022). Single-node attacks for fooling graph neural networks. *Neurocomputing*, 513:1–12.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org.

Gou, J., Yuan, X., Xue, Y., Du, L., Yu, J., Xia, S., and Zhang, Y. (2023). Discriminative and geometry-preserving adaptive graph embedding for dimensionality reduction. *Neural Networks*, 157:364–376.

Guo, W., Tondi, B., and Barni, M. (2022). An overview of backdoor attacks against deep neural networks and possible defences. *IEEE Open Journal of Signal Processing*, 3:261–287.

Han, B., Wei, Y., Wang, Q., and Wan, S. (2023). Dual adaptive learning multi-task multi-view for graph network representation learning. *Neural Networks*, 162:297–308.

Jin, H. and Zhang, X. (2019). Latent adversarial training of graph convolution networks. In *ICML workshop on learning and reasoning with graph-structured representations*, volume 2.

Jin, W., Derr, T., Wang, Y., Ma, Y., Liu, Z., and Tang, J. (2021). Node similarity preserving graph convolutional networks. In *Proceedings of the Fourteenth ACM International Conference on Web Search and Data Mining*, pages 148–156.

Jin, W., Li, Y., Xu, H., Wang, Y., and Tang, J. (2020a). Adversarial attacks and defenses on graphs: A review and empirical study. *CoRR*, abs/2003.00653.

Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., and Tang, J. (2020b). Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 66–74.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the fifth International Conference on Learning Representations*. OpenReview.net.

Li, K., Liu, Y., Ao, X., Chi, J., Feng, J., Yang, H., and He, Q. (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*, pages 925–935.

Li, Y., Jin, W., Xu, H., and Tang, J. (2020). Deeprobust: A pytorch library for adversarial attacks and defenses. *CoRR*, abs/2005.06149.

Liu, X., Jin, W., Ma, Y., Li, Y., Liu, H., Wang, Y., Yan, M., and Tang, J. (2021). Elastic graph neural networks. In *Proceedings of the thirty-eighth International Conference on Machine Learning*, volume 139, pages 6837–6849.

Liu, Z., Feng, C., Chen, S., and Hu, J. (2023). Knowledge-preserving continual person re-identification using graph attention network. *Neural Networks*, 161:105–115.

Majumdar, A. (2020). Graph transform learning. *Neural Networks*, 128:248–253.

Salha-Galvan, G., Lutzeyer, J. F., Dasoulas, G., Hennequin, R., and Vazirgiannis, M. (2022). Modularity-aware graph autoencoders for joint community detection and link prediction. *Neural Networks*, 153:474–495.

Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B., , and Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, 29(3):93–106.

Song, Y., Ye, H., Li, M., and Cao, F. (2022). Deep multi-graph neural networks with attention fusion for recommendation. *Expert Systems With Applications*, 191:116240.

Tang, X., Li, Y., Sun, Y., Yao, H., Mitra, P., and Wang, S. (2020). Transferring robustness for graph neural network against poisoning attacks. In *Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining*, pages 600–608.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *Proceedings of the sixth International Conference on Learning Representations*. OpenReview.net.

Velickovic, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. (2019). Deep graph infomax. In *Proceedings of the 7th International Conference on Learning Representations*. OpenReview.net.

Wang, Y., Wang, Y., Zhang, Z., Yang, S., Zhao, K., and Liu, J. (2023). User: Unsupervised structural entropy-based robust graph neural network. *arXiv preprint arXiv:2302.05889*.

Waniek, M., Michalak, T. P., Wooldridge, M. J., and Rahwan, T. (2018). Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2):139–147.

Wu, F., Jr., A. H. S., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Q. (2019a). Simplifying graph convolutional networks. In *Proceedings of the thirty-sixth International Conference on Machine Learning*, volume 97, pages 6861–6871.

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., and Zhu, L. (2019b). Adversarial examples for graph data: Deep insights into attack and defense. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 4816–4823. ijcai.org.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24.

Xu, H., Ma, Y., Liu, H., Deb, D., Liu, H., Tang, J., and Jain, A. K. (2020). Adversarial attacks and defenses in images, graphs and text: A review. *Int. J. Autom. Comput.*, 17(2):151–178.

Yang, Y., Rao, Y., Yu, M., and Kang, Y. (2022). Multi-layer information fusion based on graph convolutional network for knowledge-driven herb recommendation. *Neural Networks*, 146:1–10.

Yin, Y., Meng, F., Su, J., Zhou, C., Yang, Z., Zhou, J., and Luo, J. (2020). A novel graph-based multi-modal fusion encoder for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3035. Association for Computational Linguistics.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. (2020). Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823.

Yu, D., Zhang, R., Jiang, Z., Wu, Y., and Yang, Y. (2020). Graph-revised convolutional network. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2020*, volume 12459, pages 378–393. Springer.

Yu, W., Chang, T., Guo, X., Wang, M., and Wang, X. (2021). An interaction-modeling mechanism for context-dependent text-to-sql translation based on heterogeneous graph aggregation. *Neural Networks*, 142:573–582.

Zhang, A. and Ma, J. (2020). Defensevgae: Defending against adversarial attacks on graph data via a variational graph autoencoder. *CoRR*, abs/2006.08900.

Zhang, C., Xue, S., Li, J., Wu, J., Du, B., Liu, D., and Chang, J. (2023). Multi-aspect enhanced graph neural networks for recommendation. *Neural Networks*, 157:90–102.

Zhang, H. and Bai, L. (2023). Few-shot link prediction for temporal knowledge graphs based on time-aware translation and attention mechanism. *Neural Networks*, 161:371–381.

Zhang, X. and Zitnik, M. (2020). Gnnguard: Defending graph neural networks against adversarial attacks. In *Proceedings of the 33th International Conference on Neural Information Processing Systems*, volume 33, pages 9263–9275.

Zhang, Z. and Wang, B. (2023). Graph spring network and informative anchor selection for session-based recommendation. *Neural Networks*, 159:43–56.

Zhao, Y., Wang, L., Wang, C., Du, H., Wei, S., Feng, H., Yu, Z., and Li, Q. (2022). Multi-granularity heterogeneous graph attention networks for extractive document summarization. *Neural Networks*, 155:340–347.

Zhu, D., Zhang, Z., Cui, P., and Zhu, W. (2019). Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1399–1407.

Zügner, D., Akbarnejad, A., and Günnemann, S. (2018). Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856. ACM.

Zügner, D. and Günnemann, S. (2019). Adversarial attacks on graph neural networks via meta learning. In *7th International Conference on Learning Representations*. OpenReview.net.