Adaptive Multi-Space Defense Framework Against Adversarial Attacks

Xiaohui Yu[1] and Qiao Yan[1] (✉)

Shenzhen University, Shenzhen, China `iamyuxh@gmail.com` `yanq@szu.edu.cn`

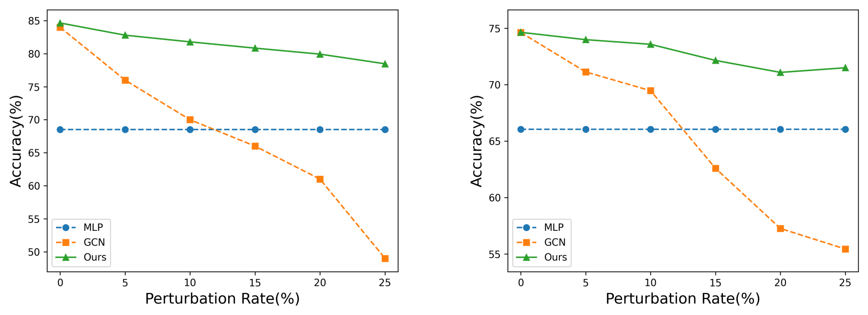
Abstract. Graph Neural Networks (GNNs) are vulnerable to adversarial attacks, leading to a significant performance degradation. Many current methods guide graph purification or graph structure learning through predefined robust properties. However, attackers can also apply the same constraints to these properties, rendering the defenses ineffective. This paper proposes an adaptive multi-sapce defense framework that enhances the robustness of GNNs without relying on prior knowledge. The core idea is to generate an estimated graph using clean attribute information and then apply graph convolution to both the perturbed graph and the estimated graph to obtain their respective node embeddings. Common embeddings between the estimated graph and the perturbed graph is then captured through shared parameters, and an attention mechanism is utilized to learn the weights of the three spaces. Extensive experiments demonstrate that our method extracts the information most relevant to classification performance where both attack methods and perturbation rates are unknown, resulting in significant improvements in both classification accuracy and performance stability.

Keywords: Adversarial robustness · Representation learning · Graph convolutional networks.

1 Introduction

Graph Neural Networks (GNNs) effectively combine attribute and structural information within graphs, demonstrating superior performance in node classification, graph classification, and link prediction tasks. They are widely applied to recommender systems [1], traffic networks [2], and financial transactions [3]. However, GNNs are vulnerable to adversarial attacks, where small perturbations mislead the model into making incorrect predictions [4]. For instance, in social networks, attackers can manipulate the network structure by adding or removing a small number of edges (social relationships), causing the GNN to misidentify communities or influence propagation paths. This vulnerability of GNNs in many critical applications can have serious consequences. Therefore, developing robust GNN models to defend against adversarial attacks is crucial.

Given the complexity of structural information, the majority of existing adversarial attacks on graph data have focused on modifying graph structure, especially adding/deleting/rewiring edges [5]. Thus, in this work, we aim to defend



(a) cora (b) citeseer

Fig. 1: Performance under Metattack with a perturbation rate of 20%

against the most common setting of adversarial attacks on graph data, i.e., poisoning adversarial attacks on graph structure.

There is a significant amount of research addressing the issue of adversarial robustness, among which defense methods based on graph purification are very effective. These methods eliminate adversarial edges or relearn the graph structure through intrinsic properties of the graph (low-rank property [11] [9], feature similarities [11] [6] [7]). However, these methods have two significant drawbacks: (1) attackers can also impose the same constraints on these predefined robust properties during the attack process, rendering the defense ineffective; (2) a large amount of valid information still exists in the perturbed original graph, which is very important for GNNs, but these methods ignore the structure of the original graph. To address these issues, we attempt to design an end-to-end defense method that does not rely on artificially defined properties and can retain the valid information of the original graph.

Figure 1 shows the node classification accuracy of GCN and MLP on the Cora and Citeseer datasets at different perturbation rates. When the perturbation rate is low, the performance of GCN is significantly higher than that of MLP, indicating that there is still a large amount of useful information in the structural information despite being contaminated. When the perturbation rate is very high, MLP, which only uses the attribute information of the graph data, performs better than GCN in classification. Therefore, when the perturbation rate is unknown, classification performance may be related to the original graph, the attribute graph, or a combination of them. Therefore, we designed an adaptive multi-space defense framework that uses graph convolutional networks to automatically capture task-relevant node representations in the graph and can adaptively adjust the weights of the original graph, attribute graph, and combined parts regardless of the perturbation rate. The results in the figure show that the method in this paper can maintain excellent performance at all perturbation rates.

Specifically, we proposed an **A**daptive multi-**S**pace defense framework, ASGCN. We use MLP to generate an estimated graph from clean node attributes. Then, we use two graph convolution modules to extract personalized representations from the two graphs that are beneficial for classification. Considering the common features between the two graphs, we designed a general convolution module with a shared parameter strategy to extract the common embeddings they share. We further utilize an attention mechanism to automatically learn the importance weights of different embeddings, thereby adaptively fusing them. In this way, node labels can supervise the learning process, adaptively adjust weights, and extract information most relevant to task performance.

Our contributions can be summarized as follows:

(1) We investigated the contributions of the original perturbed graph and the attribute graph under different perturbation rates, and concluded that task performance may be related to the perturbed graph, the attribute graph, or a combination thereof.

(2) We proposed an adaptive multi-space defense framework, ASGCN. Combined with an attention mechanism, two specific GCNs and a common GCN extract task-relevant information from the original perturbed graph, the attribute estimation graph, and their combination, respectively.

(3) Our method does not rely on manually defined graph intrinsic properties as prior knowledge, enabling end-to-end learning.

(4) Our extensive experiments on a series of benchmark datasets clearly show that MS-GCN outperforms the state-of-the-art GCNs. Furthermore, when the graph is heavily poisoned, MS-GCN can still maintain excellent classification performance.

2 Related Work

In recent years, adversarial defense research for GNNs has received increasing attention. Graph purification-based defense methods identify normal or adversarial edges based on the intrinsic properties of the graph. Jaccard-GCN [6] and GNNGuard [7] rely on the homophily assumption, with the former calculating the Jaccard similarity between node pairs and removing edges below a certain threshold, and the latter using cosine similarity to filter adversarial edges during message aggregation. STABLE [8] utilizes a similar idea of calculating similarity based on unsupervised representations instead of features. SVD-GCN [9] recognizes that Nettack [10] tends to attack high-frequency components in graph data. Therefore, it replaces the adjacency matrix with a low-rank approximation, reducing the impact of adversarial attacks while preserving important information about the structure of the graph. Graph structure learning-based methods impose regularization terms based on inherent graph properties, constraining the generated graph structure to conform to standard patterns. ProGNN [11] discovers that real-world graphs are often low-rank and sparse, with adjacent nodes having similar features. Consequently, it uses these three properties as regularization terms to constrain graph structure generation. TGNN [12] points out that previous methods ignore the connections and balance between different graph properties, proposing a tensor-based GNN framework that fuses multiple properties. RGCN [19] also relies on the intrinsic properties of the graph to adaptively weight edges, using low weights to penalize adversarial edges. DualRGNN [13] incorporates a node-similarity-preserving graph refining (SPGR) module, where these node representations contain the similarity relationships of the original nodes, thereby weakening the poisoning effect of graph adversarial attacks on graph data. Adversarial training does not rely on prior knowledge, but GOOD-AT [20] points out that it can lead to models learning incorrect information. It uses adversarial examples to define an out-of-distribution (OOD) detector as a classifier to optimize the graph structure. D4A [14] proposes smooth-less message passing to enhance the tolerance with respect to structure perturbations.

3 Preliminaries

3.1 Notations

We consider an undirected graph $G = (V, E)$, where V is the set of N nodes, where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_{ij}\}$ are the set of nodes and edges. We use matrix $A \in \{0,1\}^{N \times N}$ to denote the adjacency matrix of G. Furthermore, we use $X = [x_1, x_2, \ldots, x_n] \in R^{N \times d}$ to denote the node feature matrix where d is the dimension of the node feature vectors. $Y = [y_1, y_2, \ldots, y_n]$ are the labels corresponding to each node. In this work, we focus on the semi-supervised node classification, where the model f_θ is trained with labeled nodes $V_L \subset V$ to classify the other unlabeled nodes $V_U = V/V_L$.

3.2 A General Form of poisoning attacks

In poisoning attacks, attackers minimize an attack loss to cause incorrect final classification results. A general form of the objective for adversarial attacks can be stated as:

$$\arg \min_{\mathcal{A} \in \phi(\mathcal{A})} L_{attack}(f(\mathcal{A}, X; \theta^*), y) \quad \text{s.t.} \quad \theta^* = \arg\min_{\theta} L_{predict}(f(\mathcal{A}, X; \theta), y) \tag{1}$$

Where y denotes ground-truth labels, L_{attack} denotes the attacker's loss function, and L_{predict} denote GNN's loss. A' is the perturbed adjacency matrix, and $\phi(A)$ is a set of adjacency matrices that satisfy the unnoticeability: $\|A' - A\|_0 \leq \Delta$ in which Δ is budget to constrain the number of perturbed edges.

4 Methodology

The overall architecture of the proposed ASGCN framework is shown in Fig. 2.

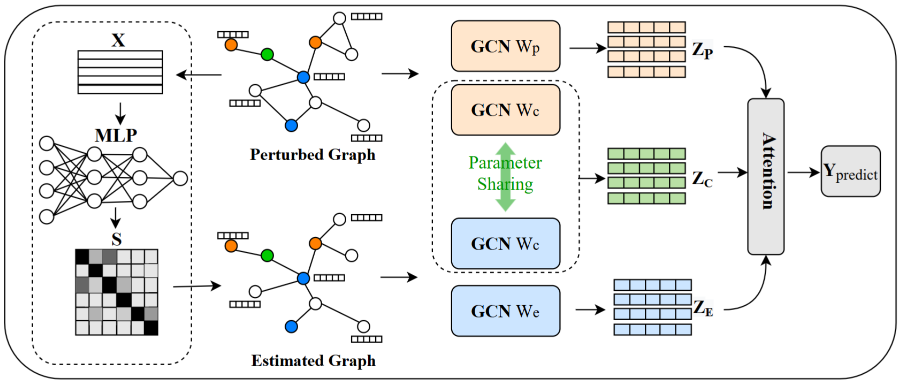


Fig. 2: The framework of the proposed ASGCN. (1) Generating an estimated graph from clean attributes using an MLP. (2) Extracting embeddings Z_P, Z_E, and Z_C from the perturbed graph, estimated graph, and a combination respectively, using two individual GCNs and one common GCN. (3) The attention mechanism assigns weights to the three spaces.

4.1 Attribute-Estimated Graph Generation

After a graph suffers a structural attack, its edges are perturbed, leading to significant noise. However, attribute information remains unaffected. We apply a multi-layer perceptron (MLP) to generate an estimated graph. The $l-th$ layer of the MLP is defined as:

$$Z_m^{(l)} = \sigma\left(Z_m^{(l-1)} W_m^{(L)}\right) \tag{2}$$

Where $Z_m^{(0)} = X$, $W_m^{(l)}$ is the learnable weight matrix for MLP, and σ is activation function. Denote the output of the final layer of MLP as Z_m, then we can obtain the soft assignment matrix $B \in R^{n\times C}$ as follows:

$$B = \text{ soft } \max(Z_m) \tag{3}$$

The element in matrix B represents the probability that the $i-th$ node belongs to class c. All parameters θ_{mlp} of the MLP are optimized using the labels from the dataset, and the loss function is defined as follows:

$$\theta^*_{mlp} = \arg\min_{\theta_{mlp}} L_{mlp} = \arg\min_{\theta_{mlp}} \frac{1}{|V_L|} \sum_{v_a \in V_L} J(b_a^{mlp}, y_a) \tag{4}$$

where b_a^{mlp} is the predicted labels of node v_a by MLP. Based on matrix B, the probability of two nodes belonging to the same class can be calculated as follows:

$$S = BB^T \tag{5}$$

The element S_{ij} in matrix $S \in R^{n\times n}$ represents the probability that node i and node j belong to the same class. Finally, we set a threshold τ to generate the estimated graph G_e as follows:

$$G_e(ij) = \begin{cases} 0, \, S_{ij} < \tau, \\ 1, \, S_{ij} \geq \tau, \end{cases} \tag{6}$$

When the value of S_{ij} is greater than τ an edge is formed in the estimated graph. After calculating this for all node pairs, we obtain the adjacency matrix A_e corresponding to the estimated graph, while the feature matrix remains X.

4.2 Specific Convolution Module

Convolutional graph neural networks obtain high-quality node representations by aggregating neighbor information. Both the perturbed graph and the estimated graph contain substantial information beneficial for node classification. In this paper, we use two GCNs [16] to obtain node representations from the perturbed graph and the estimated graph, respectively. The $l-th$ layer output in the estimated graph can be represented as

$$Z_e^{(l)} = ReLU(\widetilde{D}_e^{-\frac{1}{2}}\widetilde{A}_e\widetilde{D}_e^{-\frac{1}{2}}Z_e^{(l-1)}W_e^{(l)}) \tag{7}$$

Here, $\widetilde{A}_e = A_e + I$ is the adjacency matrix of the estimated graph G_e with added self-connections. I is the identity matrix, $\widetilde{D}_e$ is the diagonal degree matrix of $\widetilde{A}_e$, where $\widetilde{D}_e(ii) = \sum_j \widetilde{A}_e(ij)$. $W_e^{(l)}$ is the weight matrix of the $l-th$ layer in GCN. $RuLU$ is the activation function and $Z_e^{(0)} = X$. Following the same calculation method, the output embedding for the perturbed graph after graph convolution is:

$$Z_p^{(l)} = ReLU(\widetilde{D}_p^{-\frac{1}{2}}\widetilde{A}_p\widetilde{D}_p^{-\frac{1}{2}}Z_p^{(l-1)}W_p^{(l)}) \tag{8}$$

4.3 Common Convolution Module

Perturbed graph and estimated graphs contain a significant amount of correlated information relevant to node classification tasks, and this correlation is often difficult to know in advance. Therefore, in addition to capturing the individual representations of perturbed graph and estimated graph, it is also necessary to capture the common information present in both graphs. We designed a Common-GCN with a parameter sharing strategy to obtain an embedding shared between the two graphs. First, we utilize Common-GCN to extract the node embedding $Z_{cp}^{(l)}$ from perturbed graph (A_p, X) as follows:

$$Z_{cp}^{(l)} = ReLU(\widetilde{D}_p^{-\frac{1}{2}}\widetilde{A}_p\widetilde{D}_p^{-\frac{1}{2}}Z_{cp}^{(l-1)}W_c^{(l)}) \tag{9}$$

Where $W_C^{(l)}$ is the $l-th$ layer weight matrix of Common-GCN, and $Z_{cp}^{(l)}$, $Z_{cp}^{(l-1)}$ are the node representations of the perturbed graph in the $l-th$ and $(l-1)-th$

layers of Common-GNN, respectively. For estimated graph (A_e, X), we share the same weight matrix $W_C^{(l)}$ for every layer of Common-GCN as follows:

$$Z_{ce}^{(l)} = ReLU(\widetilde{D}_e^{-\frac{1}{2}} \widetilde{A}_e \widetilde{D}_e^{-\frac{1}{2}} Z_{ce}^{(l-1)} W_c^{(l)}) \tag{10}$$

Then we can get the $l-th$ common embedding $Z_c^{(l)}$ of the two graphs is:

$$Z_c^{(l)} = \frac{Z_{cp}^{(l)} + Z_{ce}^{(l)}}{2} \tag{11}$$

Sharing weights allows the two networks to focus more on the common information in the data. The node representations obtained from the common convolutional module can filter out the shared characteristics from the two graphs.

4.4 Attention Mechanism

We now have two specific embeddings Z_P and Z_E, and one common embedding Z_C. Under different perturbation rates, the contributions of the three inputs to the classification performance are different. We use the attention mechanism $att(Z_P, Z_E, Z_C)$ to learn their corresponding importance $(\lambda_p, \lambda_e, \lambda_c)$ as follows:

$$(\lambda_p, \lambda_e, \lambda_c) = att(Z_P, Z_E, Z_C) \tag{12}$$

Here $a_p, a_e, a_c \in \mathbb{R}^{n \times 1}$ indicate the attention values of n nodes with embeddings Z_P, Z_E, Z_C respectively. Taking the embedding $Z_P$in the perturbed graph as an example, the representation of node i is $z_p^i \in \mathbb{R}^{1 \times n}$. First, a non-linear transformation is performed. Then, the node attention value ξ_p^i is obtained after the operation with the attention vector $q \in \mathbb{R}^{h \times 1}$. The calculation method is as follows:

$$\xi_p^i = q^T \cdot \tanh(W \cdot (z_p^i)^T + b) \tag{13}$$

Using the method in formula (13), we can successively obtain the attention coefficients ξ_p^i, ξ_e^i of node i in Z_P, Z_E, Z_C. Using the softmax function to normalize, we obtain the weights $\lambda_p^i, \lambda_e^i, \lambda_c^i$ of the node i in Z_P, Z_E, Z_C. After the above calculation steps, each node obtains the weights on each channel. The embedding on the three channels are weighted and summed to obtain the final representation of the node:

$$Z = \lambda_p \cdot Z_P + \lambda_E \cdot Z_E + \lambda_C \cdot Z_C \tag{14}$$

4.5 Classifier

The final node representations obtained from the previous three parts are used for node classification. We employ the softmax function to compute the predicted label probabilities for each node and use the cross-entropy loss to optimize the model parameters.

$$\hat{y}_i = \text{softmax}\,(Z) \tag{15}$$

$$L_{loss} = -\frac{1}{|V_L|} \sum_{v_i \in V_L} Y_i \log(\hat{y}_i) \tag{16}$$

Y_i, $\hat{y}_i \in R^{|c|}$ is the one-hot embedding of label and the predicted label of i respectively.

5 Experiments

5.1 Experimental Setup

Dataset. We conduct extensive experiments on four widely used [11] [13] [8] graph datasets, including Cora, Citeseer, Pubmed [17], and Polblogs [18]. The Cora, Citeseer, and Pubmed 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 [13] [11], 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

Datasets	Nodes	Edges	Features	Classes	Feature type
Cora	2485	5096	1433	7	Binary
Citeseer	2110	3668	3703	6	Binary
Pubmed	19717	44338	500	3	Continuous
Polblogs	1222	16714	/	2	/

Baseline. 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:

- **GCN-Jaccard [6].** GCN-Jaccard purifies the graph structure by calculating the Jaccard similarity of node features and removing edges below a similarity threshold.
- **RGCN [19].** RGCN models the $l-th$ layer hidden representation of nodes as a Gaussian distribution and applies an attention mechanism to penalize nodes with high variance.
- **GCN-SVD [9].** Enhancing the robustness of GCNs by low-rank approximation of perturbed graphs is also a strategy to defend against adversarial attacks through preprocessing.
- **GNNGuard [7].** GNNGuard employs the theory of network homophily to assign higher scores to edges connecting similar nodes while pruning edges between unrelated nodes.

– **ProGNN [11].** ProGNN generates a clean graph structure using three constraints: low-rank, sparsity, and feature similarity of neighboring nodes.
– **STABLE [8].** STABLE utilizes a similar idea of calculating similarity based on unsupervised representations instead of features.
– **Good-AT [20].** Good-AT defines an OOD detector as a classifier to filter adversarial edges using adversarial examples.

Parameter settings. We randomly choose 10% of nodes for training, 10% of nodes for validation and the remaining 80% of nodes for testing. To obtain the three representations in our model, we simultaneously train three 2-layer GCNs. These GCNs share the same hidden layer dimension (nhid1) and the same output dimension (nhid2), where $nhid1 \in \{512, 768\}$ and $nhid2 \in \{32, 128, 256\}$. The MLP pre-training was performed for 100 epochs, with a single hidden layer of size nhid = nhid2. We explore weight decay in $\{5e-3, 5e-4\}$ and use a dropout rate of 0.5. The learning rate is searched within the range of 0.01 to 0.05. τ is an important parameter in the ASGCN. Too large a value will cause the estimated graph to be too dense, while too small a value will lead to underlearning. In this paper, based on the number of edges in each dataset, the (n * number of edges) node pairs with the greatest similarity, S_{ij}, are chosen to form edges. n is searched in$\{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$. To ensure a fair comparison with the baseline methods, we adopt the same set of hyperparameters as recommended by their respective authors across all our models. All results were performed five times, and the results were averaged.

5.2 Performance on clean graph

Excellent defense methods should not only demonstrate robustness on perturbed graphs but also maintain comparable performance on clean graph data. Table 2 shows the classification results of baselines and our proposed method on clean datasets. The results demonstrate that our method maintains competitive performance across all four datasets, indicating that it does not sacrifice excessive original data accuracy in order to improve robustness against adversarial attacks. This is because, even when the data is unperturbed, our method can adaptively adjust the weights of the three representations to obtain the most task-relevant information.

Table 2: Performance on clean graph. The top two performances are highlighted in **bold** and underline.

Datasets	GCN-Jaccard	RGCN	GCN-SVD	GNNGuard	ProGNN	STABLE	Good-AT	Ours
Cora	82.45±0.32	83.09±0.15	80.63±0.46	79.51±0.30	83.45±0.41	83.54±0.40	84.30±0.16	**84.65±0.15**
Citeseer	72.43±0.24	71.20±0.31	70.65±0.19	71.83±0.45	73.87±0.31	73.95±0.37	74.08±0.28	**74.64±0.12**
PubMed	85.06±0.08	85.16±0.06	83.44±0.14	83.88±0.05	**87.24±0.13**	85.79±0.04	84.14±0.08	86.24±0.06
Polblogs	–	94.79±0.15	**95.06±0.20**	-	94.68±0.19	94.96±0.07	93.79±0.20	94.46±0.16

Table 3: Node classification performance under non-targeted attack(metattack). The top two performances is highlighted in **bold** and underline.

Datasets	Ptb Rate	GCN-Jaccard	RGCN	GCN-SVD	GNN Guard	ProGNN	STABLE	Good-AT	Ours
Cora	5%	79.13	77.42	78.39	78.27	82.27	81.49	79.82	**82.80**
	10%	75.16	72.22	71.47	78.03	79.03	80.06	70.26	**81.79**
	15%	71.03	66.82	66.69	78.18	76.40	78.45	67.80	**80.84**
	20%	65.71	59.27	58.94	77.15	73.32	78.01	55.60	**79.93**
	25%	60.82	50.51	52.06	76.34	69.72	71.22	51.23	**78.47**
	Avg.	70.37	65.25	65.51	77.59	76.15	77.85	64.94	**80.77**
Citeseer	5%	70.51	70.50	68.84	71.32	73.09	73.61	72.39	**73.99**
	10%	69.54	67.71	68.87	70.86	72.51	73.13	69.46	**73.58**
	15%	65.95	65.69	63.26	70.83	72.03	72.13	66.78	**72.15**
	20%	59.30	62.49	58.55	70.97	70.02	72.43	60.27	**71.09**
	25%	59.89	55.35	57.18	71.08	68.95	70.21	57.90	**71.50**
	Avg.	65.04	64.35	63.34	71.01	71.32	72.30	65.36	**72.46**
PubMed	5%	85.44	82.51	83.64	84.24	87.23	**87.32**	84.02	86.36
	10%	85.26	80.36	82.31	84.10	**87.21**	87.16	83.87	85.82
	15%	84.72	75.84	82.26	84.14	87.20	**87.24**	83.07	85.75
	20%	83.65	71.24	83.07	84.22	**87.15**	87.04	82.41	85.18
	25%	83.74	70.12	81.88	84.12	86.76	**87.15**	82.00	85.54
	Avg.	84.56	76.01	82.63	84.16	87.11	**87.18**	83.07	85.73
Polblogs	5%	-	74.34	89.09	-	93.29	93.21	91.45	**93.33**
	10%	-	71.04	81.24	-	89.42	**92.14**	87.26	91.92
	15%	-	67.28	68.10	-	86.04	90.23	80.88	**93.76**
	20%	-	59.86	57.33	-	79.56	88.43	74.14	**92.84**
	25%	-	56.02	48.66	-	63.18	84.56	69.51	**90.69**
	Avg.	-	65.71	68.88	-	82.30	89.71	80.65	**92.51**

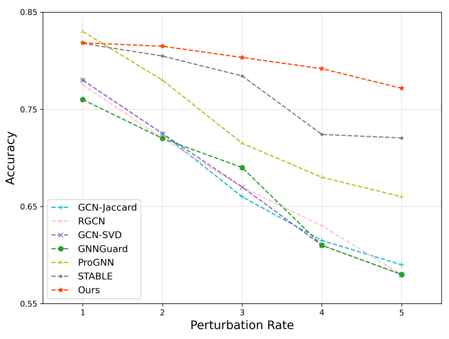
5.3 Defense Performance

5.3.1 Against Non-targeted Adversarial Attacks The goal of non-targeted attack is to degrade the overall performance of GNNs on the whole graph.We use Mettack [21] to perform non-targeted poisoning attacks. , which is an effective attack method. Mettack needs to compute meta-gradients, which requires huge memory space to store the computation graphs in all iterations, so we only use the perturbed graphs provided by ProGNN [11]. We vary the perturbation rate, i.e., the ratio of changed edges, from 0 to 25% with a step of 5%. Due to the node features being unavailable on the Polblogs dataset, it is no means of evaluating the baselines methods Jaccard-GCN and GNNGuard on this dataset.The defense results are shown in Table 3, we have the following observations:

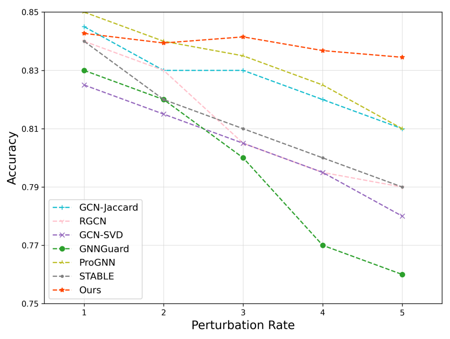
– Our method consistently outperforms all baseline methods across different perturbation rates on the Cora, Citeseer, and Polblogs datasets, while also achieving competitive performance on the PubMed dataset.
– Our method exhibits stable performance across varying attack rates. Across the four datasets, as the perturbation rate increases from 5% to 25%, our

method's performance decreases by only 4.33%, 2.49%, 0.57%, and 0.79%, respectively. In contrast, ProGNN [11] performs well at low perturbation rates but experiences a sharp decline in performance as the perturbation rate increases. GNNGuard [7] and STABLE [8] demonstrate good defense stability, but their accuracy is lower than that of our method. This indicates that our method is insensitive to the perturbation rate and can adaptively regulate the coefficients of the three convolutional layers in scenarios where the perturbation rate is unknown, consistently identifying the most beneficial components for classification.

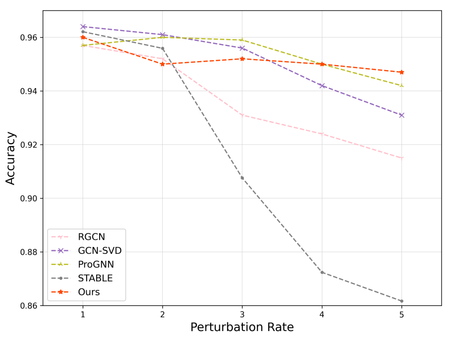
– Even when the graph is heavily poisoned, our method maintains superior classification performance. For example, on the Cora dataset at a 25% perturbation rate, our method achieves a node classification accuracy of 78.47%, significantly higher than Good-AT [20] (51.23%) and RGCN [19] (50.51%). This demonstrates the robustness of our approach under varying levels of adversarial perturbations.



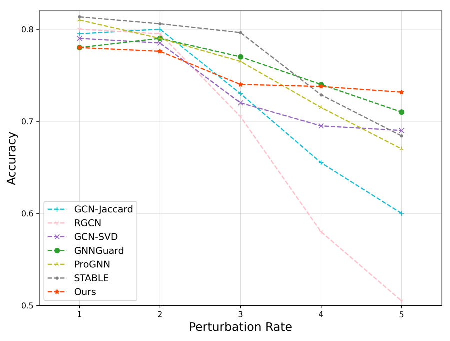
(a) cora



(b) pubmed



(c) polblogs



(d) citeseer

Fig. 3: Results of different models under nettack

5.3.2 Against Targeted Adversarial Attacks Targeted attack generates attacks on specific nodes and aims to fool GNNs on these target nodes. we employ Nettack [10] as the target graph adversarial attack method, and we follow the default parameter settings in the authors' original implementation. Following [11] [13], 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. Figure 3 presents the node classification results under different perturbation rates. From the results, we can make a few observations as follows:

– On the Cora and Pubmed datasets, our ASGCN method can effectively defend against targeted graph adversarial attacks, outperforming all baseline methods in most cases. Compared to the ProGNN [11] and STABLE [8] methods, ASGCN achieves higher semi-supervised node classification accuracy. Furthermore, at perturbation rates of 4 and 5, ASGCN maintains performance of 79.18% and 77.16% on the Cora dataset, significantly exceeding other baseline methods.
– On the Citeseer and Polblogs datasets, ASGCN does not outperform certain baseline methods on perturbed graphs. However, the overall performance of ASGCN remains comparable to most baseline methods, particularly when the perturbation rate is 5, where ASGCN still exhibits the best performance.

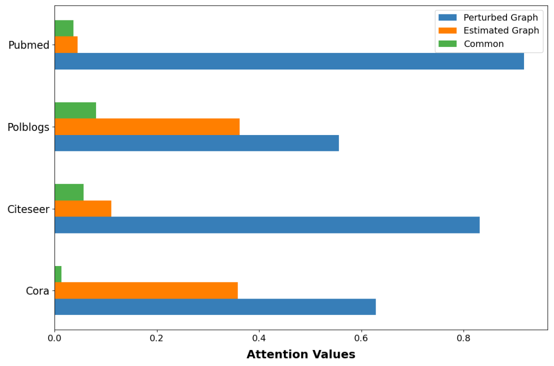
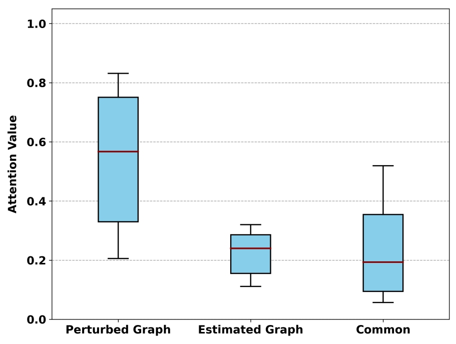


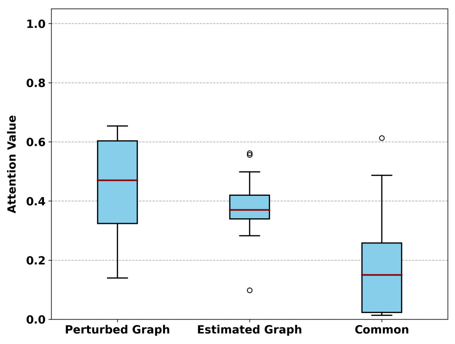
Fig. 4: Attention values under metattack (20% perturbation)

5.4 Analysis of Attention Mechanism

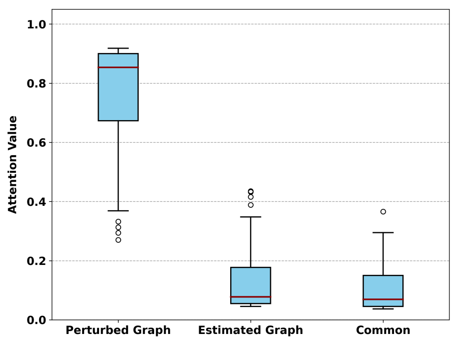
To investigate our proposed model's ability to adaptively learn the weights assigned to the perturbed graph, the estimated graph, and the common space – under perturbations with arbitrary attack rates – we observed the attention value distribution across these three components for all datasets under metattack with



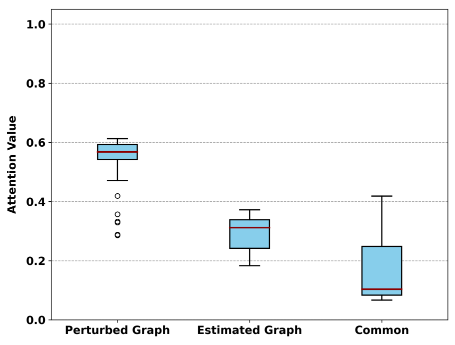
(a) citeseer



(b) cora



(c) pubmed



(d) polblogs

Fig. 5: Attention distribution under metattack with a perturbation rate of 20%

a 20% perturbation rate. As illustrated in Figure 5, the attention distribution differs for the three spaces in each dataset. For the PubMed dataset, the information contribution from the perturbed graph is significantly greater than that of the other two spaces. In the Cora dataset, the contributions of the perturbed graph and the estimated graph are comparable.

Furthermore, Figure 4 illustrates the attention values for the three spaces (original graph, estimated graph, and common space) when each dataset achieves its best classification accuracy under Metattack with a 20% perturbation rate. In Pubmed, the original perturbed graph still provides the most useful information. However, in Cora and Polblogs, the attention values for the estimated graph increase. Across all datasets, the common space values are the lowest, but they are also distinct for each dataset. In summary, the experiment demonstrates that our proposed ASGCN is able to adaptively assign a larger attention value to the more important information.

6 Conclusion

In this paper, we investigate the limitations of methods relying on predefined robust properties. To address these limitations, we propose an adaptive multi-space defense framework that leverages the original graph structure to varying degrees. By incorporating an attention mechanism, two specific convolution modules, and one conmon convolution module, we can capture task-relevant information. Through extensive experiments, we validate the adversarial robustness of ASGCN against poisoning attacks. Furthermore, even when the graph is heavily poisoned, ASGCN can still maintain excellent performance.

References

1. Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H., He, Q.: A survey on knowledge graph-based recommender systems. IEEE Transactions on Knowledge and Data Engineering. 34(8),3549(2021)
2. Jiang, W., Luo, J.: Graph neural network for traffic forecasting: A survey.Expert Systems with Applications. 207(2022)
3. Wang, J., Zhang, S., Xiao, Y., Song, R.: A review on graph neural network methods in financial applications. CoRR abs/2111.15367.(2021)
4. Xu, K., Chen, H., Liu, S., Chen, P., Weng, T., Hong, M., Lin, X. : Topology attack and defense for graph neural networks: An optimization perspective. In: International joint conference on artificial intelligence, pp. 3961–3967(2019)
5. Han Xu,Yao Ma,Hao chen Liu,Debayan Deb,Hui Liu.: Adversarial attacks and defenses in images,graphs and text:A review. arXiv preprint arXiv:1909.08072(2019).
6. Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, Liming Zhu. Adversarial Examples on Graph Data: Deep Insights Into Attack and Defense. In: International Joint Conference on Artificial Intelligence(2019)
7. Xiang Zhang, Marinka Zitnik.: GNNGuard: Defending Graph Neural Networks Against Adversarial Attacks. In: Neural Information Processing Systems(2020).
8. Li, K., Liu, Y., Ao, X., Chi, J., Feng, J., Yang, H., He, Q.: 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. pp.925–935(2022)
9. N. Entezari, S.A. Al-Sayouri, A. Darvishzadeh, E.E. Papalexakis: All you need is low (rank): Defending against adversarial attacks on graphs. In: International Conference on Web Search and Data Mining(2020)
10. Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann.: Adversarial attacks on neural networks for graph data. In: ACM SIGKDD(2018)
11. Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang.: Graph Structure Learning for Robust Graph Neural Networks. In:ACM SIGKDD(2020)
12. Jianfu Zhang, Yan Hong, Dawei Cheng, Liqing Zhang, and Qibin Zhao.: Defending Adversarial Attacks in Graph Neural Networks Via Tensor Enhancement, PATTERN RECOGNITION, 158(2025)
13. Qian T, Jianpeng L, Enze Z, Lusi L, et al.: A Dual Robust Graph Neural Network Against Graph Adversarial Attacks, Neural networks : the official journal of the International Neural Network Society, 175(2024)

14. Li X, Gan Z, Bai Y, et al. D4A: An efficient and effective defense across agnostic adversarial attacks[J]. Neural Networks, 183(2025)
15. Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski.: Are defenses for graph neural networks robust? In: Advances in Neural Information Processing Systems. 35(2022)
16. Thomas N. Kipf, Max Welling. Semi-supervised classification with graph convolutional networks.In: International Conference on Learning Representations(2017)
17. Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine, 29(3), 93–106(2008)
18. Adamic, L. A., Glance, N.: The political blogosphere and the 2004 U.S. election: Divided they blog. New York, NY, USA: Association for Computing Machinery.(2005)
19. Zhu, D., Zhang, Z., Cui, P., Zhu, W. Robust graph convolutional networks against adversarial attacks. In: ACM SIGKDD, pp.1399–1407(2019)
20. Kuan Li, YiWen Chen, Yang Liu, Jin Wang, Qing He, Minhao Cheng, and Xiang Ao.: Boosting the Adversarial Robustness of Graph Neural Networks: an OOD Perspective., In: International Conference on Learning Representations (2024)
21. Zügner, D., Günnemann, S.: Adversarial attacks on graph neural networks via meta learning. In: 7th international conference on learning representations.OpenReview.net(2019)