This is the Pre-Published Version.
The following publication X. Ai, Y. Zhu, Y. Zheng, G. Li, J. Li and K. Zhou, "Revisiting Adversarial Robustness of GNNs Against Structural Attacks: A Simple and Fast Approach," in IEEE Transactions on Information Forensics and Security, vol. 21, pp. 446-459, 2026 is available at https://doi.org/10.1109/TIFS.2025.3641816.

Revisiting Adversarial Robustness of GNNs against Structural Attacks: A Simple and Fast Approach

Xing Ai, Yulin Zhu, Yu Zheng, Gaolei Li, Jianhua Li, and Kai Zhou*

***Abstract*—To defend against adversarial structural attacks on graphs, we analyze attacks through the lens of mutual information and discover the "pairwise effect". This effect reveals that structural attacks effectively degrade the performance of victim GNNs when these GNNs receive the modified structure paired with the given node attributes as training input. Therefore, we propose a novel defense strategy that renders structural attacks ineffective by disrupting the pairing of modified structures and node attributes during the training of victim GNNs, which we call "disrupting the pairwise effect". To implement this idea, we propose two simple yet effective training strategies: Structural Fine-Tuning (SF) and Progressive Structural Training (PST), which disrupt the pairwise effect through node attributes pre-training followed by structure fine-tuning and progressive structure training, respectively. Compared to existing robust GNNs, our strategies avoid time-consuming techniques, thereby improving the robustness of GNNs while enhancing training speed. Additionally, these strategies can be easily applied to a wide range of commonly used GNNs, including robust GNN variants, making them highly adaptable to different models and applications. We provide theoretical analysis of the proposed training strategies and conduct extensive experiments on various datasets to demonstrate their effectiveness. Datasets and codes of this paper are available at https://github.com/Xing-Ai1003/Revisiting-Adversarial-Robustness-of-GNNs.**

***Index Terms*—Graph learning, structural attacks, robust GNN, mutual information.**

I. Introduction

GRAPH Neural Networks (GNNs) have emerged as the leading approach to graph learning tasks in various domains, including recommender systems [1], social networks [2], and bioinformatics [3]. However, numerous studies [4]–[6] have demonstrated the vulnerability of GNNs under *adversarial attacks*, where an attacker can deliberately modify graph data to cause the misprediction of GNNs. Among them, *structural attacks* [7], [8] have gained prominence due to the unique structural nature of graph data.

Specifically, by solely modifying the edges in a graph, structural attacks hold practical significance in application scenarios where attackers have limited access to the relationships among entities rather than the attributes of the entities themselves. A notable example is fraud detection in online recommendation systems [9], [10], where users and products are represented as nodes and their interactions are denoted by edges. Although a fraudulent user cannot change the basic information of other users (i.e., node attributes), they can interact with others by participating in activities like purchasing the same products or providing manipulated ratings to target products. This essentially allows the fraudulent user to alter the graph structure with the malicious intent of evading detection.

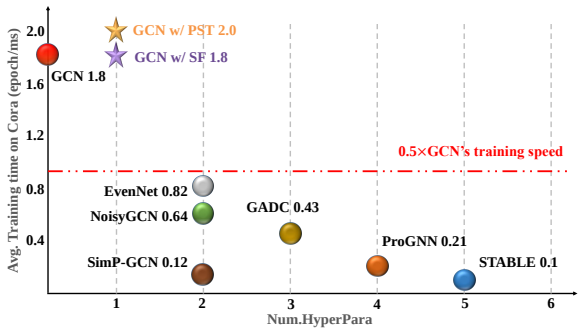


Fig. 1: Comparison of training speed (vertical axis) and extra hyperparameters number (horizontal axis) among GCN, GCN with proposed strategies (stars), and existing robust GNNs.

As a result, numerous studies have focused on developing effective defense strategies against structural attacks. A prevalent approach is to design *adversarially robust* Graph Neural Network (GNN) models [11]–[14] under attack. The main ideas behind these robust models involve purifying the modified graph structure or designing adaptive aggregation mechanisms to avoid aggregating messages through modified structures. For instance, GCN-Jaccard [12] uses the Jaccard index to identify and remove malicious edges, thereby purifying the modified structures. SimP-GCN [13] calculates the similarity of node features for each node pair within a graph and applies an adaptive aggregation strategy to balance the information from the graph structure and node features.

However, purifying the modified structure or applying adaptive aggregation imposes high computational complexity on robust GNNs. Moreover, these robust models often introduce additional hyperparameters (e.g., weighting coefficients and thresholds) beyond the basic ones (e.g., learning rates, dropout ratio, epochs). This can make the model more complex and harder to understand, while also increasing the difficulty of hyperparameter tuning.

Specifically, Fig. 1 compares robust GNNs, vanilla GCN [15], and GCN with two proposed training strategies in terms of training speed (average epoch number per millisecond) on Cora dataset and the number of extra hyperparameters.

Xing Ai and Kai Zhou are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China. E-mail: 22039684r@connect.polyu.hk, kaizhou@polyu.edu.hk. Yulin Zhu is with Department of Computer Science, Hong Kong Chu Hai College, Hong Kong. E-mail: ylzhu@chuhai.edu.hk Yu Zheng is with the University of California, Irvine, Irvine, CA, USA. E-mail: yu.zheng@uci.edu Gaolei Li and Jianhua Li are with Shanghai Jiao Tong University, Shanghai, China. E-mail: gaolei_li@sjtu.edu.cn, lijh888@sjtu.edu.cn. Corresponding author: Kai Zhou.

This work is partly supported by National Natural Science Foundation of China under Grant No. 62572314 and No. 62471301 and Hong Kong RGC Project (No. PolyU25210821).

The results show that all robust GNNs are slower than half of GCN's training speed (highlighted by the red dashed line at 0.9 epoch/ms), with some (e.g., SimP-GCN [13] and STABLE [16]) being even slower than one-tenth of GCN's speed. Additionally, these robust GNNs require at least two extra hyperparameters compared to GCN. It demonstrates that robust GNNs incur a significant cost in terms of training time and hyperparameter complexity, thereby limiting their practicality and deployability in real-world applications.

In response, we aim to investigate this fundamental research question: *Can we enhance the adversarial robustness of GNNs against structural attacks while avoiding a significant increase in training time and hyperparameter complexity?*

To this end, our first step is to analyze structural attacks *from the perspective of mutual information*. The major finding is that the essence of structural attacks lies in minimizing the conditional mutual information between the modified structures and labels, given the node attributes. This means that structural attacks significantly reduce the information that GNNs can learn, thereby degrading their performance. This essence leads to a crucial characteristic of structural attacks, namely "pairwise effect", which indicates that the attack is most effective when the modified structure is paired with the given node attributes.

This in-depth analysis offers a novel defense strategy: replacing node attributes with alternative node attributes to disrupt the pairwise effect, thus rendering structural attacks ineffective. However, directly constructing alternative node attributes is computationally intensive, due to the high dimensionality of node attributes. To disrupt pairwise effect efficiently, we propose two straightforward yet effective training strategies, **Structural Fine-Tuning (SF)** and **Progressive Structural Training (PST)**. They utilize the trained node embeddings during the GNNs training as the alternative node attributes. Specifically, SF and PST utilize node attributes pre-training and progressive structure training respectively, to obtain the trained node embeddings.

The advantages of these two training strategies include:

- Model Universality: These strategies are independent of specific model implementation details and can be integrated into a variety of GNNs without requiring significant modifications. We successfully apply these strategies in five widely used vanilla GNNs and four robust GNNs.
- Robustness: Our training strategies significantly improve the robustness of both vanilla GNNs and robust GNNs against structural attacks. For example, on the Cora dataset, under a perturbation ratio of 20% with Mettack, the accuracy of GCN is improved by 15.2% and 16.9% after applying SF and PST, respectively. For robust GNNs, SF and PST improve the accuracy of GCN-Jaccard from 75. 2% to 80.0% and 81.2%, respectively.
- Efficiency: The strategies do not introduce time-consuming components, thereby enhancing the training speed of the models. For example, PST doubles the training speed of GCN-Jaccard, which significantly alleviates the time-consuming problem. Moreover, SF improves the training speed of GCNs with a more than 100% increase on large-scale graphs [17]. As shown in Fig. 1, GCN with two proposed strategies, i.e., GCN w/ PST and GCN w/ SF, require only one extra hyperparameter and achieve comparable or faster training speed to GCN, whereas robust GNNs train at less than half the speed of GCN.

In summary, our major contributions are as follows.

1) We utilize mutual information to investigate structural attacks and highlight the "pairwise effect." It provides a novel defense idea to defend against such attacks.
2) We propose two straightforward and effective training strategies that can be applied to various GNNs to enhance their robustness and improve their training speed.
3) The comprehensive evaluation of the proposed training strategies against state-of-the-art baselines on node classification benchmarks, encompassing large-scale graph datasets, reveals that they achieve superior robustness while significantly enhancing training efficiency.

The rest of this paper is organized as follows. Sec. II and Sec. III provide overviews of the related work and background. Sec. IV defines the primary research problem of this paper. Sec. V provides theoretical analysis of the pairwise effect. Sec. VI comprehensively introduces the proposed training strategies, and Sec. VII describes our experimental setting and empirically demonstrates the proposed strategies' performance and scalability. Finally, Sec. VIII concludes the paper.

II. Related Works

A. Graph Neural Networks

The learning target of Graph Neural Networks (GNNs) is to generalize meaningful representations for the input graph. The basic technique of GNN is message passing, which aggregates attributes between nodes through the edges to capture complex relationships [18]. By leveraging connectivity interactions between nodes, GNNs update and refine node embeddings, thereby enabling a wide range of downstream tasks, including node classification [19], [20], link prediction [21], [22], and graph-level classification [23], [24] and so on. Graph Attention Network (GAT) [25] enhances the aggregation mechanism by calculating the importance of each neighbor and assigning greater attention weights to important neighbors. Hamilton et al. [26] propose GraphSAGE, an inductive framework for GNNs that conducts a sampling method to select important local structures to aggregate, thus allowing it to handle large-scale graphs. Xu et al. [27] propose Graph Isomorphism Network (GIN) for the graph classification task and indicate that three critical functions must be injective.

B. Structural Attacks in Graph Learning

Structural attacks are a popular form of attack covering a wide range of graph learning models beyond GNNs, including self-supervised learning [28], signed graph analysis [29], [30], recommendation systems [31], etc. The primary idea is to utilize gradient-based methods to search for the optimal graph structure to degrade the performance of various tasks. For instance, Mettack [7] formulated global structural poisoning attacks on GNNs as a bilevel optimization problem and

leveraged a meta-learning framework to solve it. BinarizedAttack [32] simplified graph poisoning attacks against graph-based anomaly detection to a one-level optimization problem. HRAT [33] proposed a heuristic optimization model integrated with reinforcement learning to optimize the structural attacks against Android malware detection. GraD [8] proposes a reasonable budget allocation mechanism to enhance the effects of structural attacks.

C. Robust GNNs

To defend against structural attacks, a series of robust GNNs is proposed, which typically rely on purifying the modified structure or designing adaptive aggregation strategies. For example, GNNGuard [34] removes the malicious links during training by considering the cosine similarity of node attributes. Zhao et al. [35] used a conservative Hamiltonian flow to improve the model's performance under adversarial attacks. Zhuang et al. [36] propose a dual neighborhood contrastive learning strategy to learn local topological-semantic robust representations. However, common drawbacks of these approaches include high computational overhead and hyperparameter complexity. More recently, few works have attempted to develop efficient robust GNNs. For example, NoisyGCN [37] defends against structural attacks by injecting random noise into the architecture of GCN, thereby avoiding complex strategies and improving runtime. Similarly, EvenNet [38] proposes an efficient strategy that ignores odd-hop neighbors of nodes, with a time complexity that is linear to the number of nodes and edges in the input graph. These efforts significantly reduce the time complexity of building robust GNNs but still introduce additional hyperparameters. NoisyGCN requires careful selection of the ratio of injected noise, and EvenNet requires the determination of both the order of the graph filter K and the initialization hyperparameter α. This motivates us to develop even simpler yet robust GNNs.

III. Preliminaries

In this section, we explore the foundational concepts relevant to mutual information, structural attacks, and robust GNNs. We show the notations in Table. I.

TABLE I: A lookup table of frequently used notations.

Notation	Description
$\mathcal{G}$	original graph
$\mathbf{X}$	node attributes matrix
$\mathbf{A}$	adjacency matrix
$\mathbf{Y}$	labels
$\hat{\mathbf{A}}$	perturbed adjacency matrix
$\mathbf{X}^{'}$	alternative node attributes
$\mathbf{E}$	identity matrix with same size as $\mathbf{A}$
n	the number of nodes
d	the dimension of node attributes
δ	structural perturbation
f	GNN model
θ	learnable weights of GNNs
$\mathcal{L}$	loss function
$I(\cdot;\cdot)$	mutual information
$I(\cdot;\cdot\|\cdot)$	conditional mutual information
$H(\cdot)$	information entropy
$H(\cdot\|\cdot)$	conditional information entropy

A. Graph Neural Networks and Structural Attacks

Let $\mathcal{G} = (\mathbf{X}, \mathbf{A}, \mathbf{Y})$ be an input graph with n nodes, where $\mathbf{X} \in \mathbb{R}^{n\times d}$ denotes node attributes, $\mathbf{A} \in \{0,1\}^{n\times n}$ is the adjacent matrix represents structure information, and $\mathbf{Y}$ represents the labels of the nodes. A GNN model f_θ parameterized with θ accepts ($\mathbf{X}$, $\mathbf{A}$) to output node embeddings $f_\theta(\mathbf{X}, \mathbf{A})$. θ is trained to minimize the loss function $\mathcal{L}$:

$$\theta^* = \arg\min_{\theta} \mathcal{L}(f_\theta(\mathbf{X}, \mathbf{A}), \mathbf{Y}). \tag{1}$$

Structural attacks against semi-supervised node classification naturally fit within a *poisoning attack setting*, where the GNN model is trained and makes predictions over a manipulated graph. In a worst-case scenario, it is assumed that the attacker can arbitrarily modify the graph structure (i.e., $\mathbf{A}$) to degrade classification performance. Specifically, the attacker seeks to find an optimal structural perturbation δ^*, which maliciously injects or removes edges into graph structures, resulting in a poisoned graph $\mathcal{G}' = (\mathbf{X}, \hat{\mathbf{A}} = \mathbf{A} + \delta^*, \mathbf{Y})$. Mathematically, a structural attack can be formulated as solving a bilevel optimization problem:

$$\begin{aligned} \delta^* &= \arg\min_{\delta} \mathcal{L}_{atk}(f_{\theta^*}(\mathbf{X}, (\mathbf{A}+\delta)), \mathbf{Y}) \\ \text{s.t.} \quad \theta^* &= \arg\min_{\theta} \mathcal{L}(f_\theta(\mathbf{X}, \mathbf{A}+\delta), \mathbf{Y}), \end{aligned} \tag{2}$$

where $\mathcal{L}_{atk}$ quantifies the attack objective. The attacks [4], [7], [8] mainly differ in their specific algorithms to solve the optimization problem.

B. Robust GNNs as Defense

Training robust GNN models is a common defense strategy to mitigate structural attacks. In this paper, the defender's goal is to train a robust GNN model from the poisoned graph to maintain node classification accuracy. We note that the defender only has access to the poisoned graph $\mathcal{G}' = (\mathbf{X}, \mathbf{A} + \delta^*, \mathbf{Y})$, not the clean graph $\mathcal{G}$. Additionally, the defender does not have prior knowledge about how the perturbation δ^* was generated. The defender's goal can be formulated as searching for a robust operation M^* for the inputs to let GNNs predict labels as correctly as possible:

$$M^* = \arg\min_{M} \mathcal{L}(f_\theta(M(\hat{\mathbf{A}}, \mathbf{X})), \mathbf{Y}). \tag{3}$$

Since defenders can only access the modified graph without prior knowledge of attacks, existing robust GNNs mostly implement M^* through purifying or adaptive aggregation, which are time-consuming and introduce extra hyperparameters.

C. Mutual Information

Mutual information (MI) is a measure of the amount of information shared between variables. It quantifies the reduction in uncertainty about a variable given the knowledge of another. The MI between two variables $\mathbf{X}$ and $\mathbf{Y}$ is defined as:

$$I(\mathbf{X};\mathbf{Y}) = H(\mathbf{Y}) - H(\mathbf{X}|\mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{Y}|\mathbf{X}), \tag{4}$$

where $H(\cdot)$ and $H(\cdot|\cdot)$ are the information entropy and conditional information entropy, respectively. MI is a non-negative quantity, and it is equal to zero when $\mathbf{X}$ and $\mathbf{Y}$ are

independent. The maximum value of MI is achieved when $\mathbf{X}$ and $\mathbf{Y}$ are perfectly correlated. The mutual information shared among three variables is defined as:

$$I(\mathbf{X};\mathbf{Y};Z) = I(\mathbf{X};\mathbf{Y}) - I(\mathbf{X};\mathbf{Y}|Z), \tag{5}$$

where $I(\mathbf{X};\mathbf{Y}|Z)$ is the conditional mutual information between $\mathbf{X}$ and $\mathbf{Y}$ under the given Z.

IV. Problem Statements

A. Threat Model

In this work, we focus on scenarios where attackers maliciously modify the structure of a graph by inserting or removing edges, thus degrading the performance of GNNs on downstream tasks. To evaluate the robustness of GNNs under the most adverse conditions, we consider the white-box attacks, where the attackers have complete knowledge of the victim GNN and can access the clean graph. Specifically, for the original clean graph $\mathcal{G} = (\mathbf{X}, \mathbf{A}, \mathbf{Y})$, structural attacks modify the structure and output the poisoned graph $\mathcal{G}^{'} = (\mathbf{X}, \hat{\mathbf{A}}, \mathbf{Y})$, where $\hat{\mathbf{A}}$ is the modified structure that satisfies $||\mathbf{A} - \hat{\mathbf{A}}|| < \mathcal{B}$. $\mathcal{B}$ is the budget, which limits the number of edges that attackers can modify.

B. Problem of Defense

The primary goal of defense is to enhance the robustness of GNNs against structural attacks, i.e., ensuring that the model's performance remains stable when the graph structure is modified. Defenders have access only to the poisoned graph $\mathcal{G}^{'} = (\mathbf{X}, \hat{\mathbf{A}}, \mathbf{Y})$ and do not have access to the original clean graph. Additionally, defenders have no knowledge about the attackers, including their strategies or the specific modifications they have applied. Therefore, the problem of defense is developing methods that can mitigate the effects of structural attacks without any prior knowledge of the original graph or the attackers' actions.

V. Pairwise Effect: A Mutual Information Perspective of Adversarial Structural Attacks

In this section, we first analyze structural attacks through a mutual information perspective, which reveals that the essence of structural attacks lies in minimizing the conditional mutual information. Then, we show that this essence caused a vital feature of structural attacks – *pairwise effect*, which inspires us to develop a novel defense approach: causing a mismatch between the structure and node attributes (i.e., disrupting the pairwise effect) to render structural attacks ineffective.

A. Essence of Structural Attacks

To better understand the essence of structural attacks, we first introduce mutual information to analyze GNNs. For a well-trained GNN f that can perfectly predict labels, it mainly learns information from two parts: (1) $I(\mathbf{X};\mathbf{Y})$, which is the mutual information between the node attributes and labels, (2) $I(\mathbf{A};\mathbf{Y}|\mathbf{X})$, which is the conditional mutual information between the structures and labels under the node attributes. The essence of structural attacks is seeking the optimal modified structure to minimize the latter, thus reducing the information GNNs can learn. *To ensure better readability, we introduce the analysis results below while providing all the mathematical proof in the Appendix. A.*

According to the data processing inequality [39], for a GNN f, the mutual information between node embeddings $f(\mathbf{X}, \mathbf{A})$ and labels is less than or equal to the mutual information between the joint distribution $(\mathbf{X}, \mathbf{A})$ and labels $\mathbf{Y}$:

$$I(f(\mathbf{X}, \mathbf{A}); \mathbf{Y}) \leq I((\mathbf{X}, \mathbf{A}); \mathbf{Y}). \tag{6}$$

The equality in the above inequality holds if f can capture all the information that $(\mathbf{X}, \mathbf{A})$ contains about $\mathbf{Y}$. In practice, the primary training objective of a GNN is to ensure that it can extract sufficient information from the input data, i.e. $(\mathbf{X}, \mathbf{A})$, to perfectly predict the labels $\mathbf{Y}$. The case that corresponds to this training objective is when the equality is achieved, i.e., f is a well-trained GNN capable of fully extracting information from inputs to predict labels perfectly:

$$I(f(\mathbf{X}, \mathbf{A}); \mathbf{Y}) = I((\mathbf{X}, \mathbf{A}); \mathbf{Y}). \tag{7}$$

According to the properties of mutual information, $I((\mathbf{X}, \mathbf{A}); \mathbf{Y})$ is equal to the sum of $I(\mathbf{X}; \mathbf{Y})$ and $I(\mathbf{A}; \mathbf{Y}|\mathbf{X})$, thus we can rewrite the above equation as:

$$I(f(\mathbf{X}, \mathbf{A}); \mathbf{Y}) = I((\mathbf{X}, \mathbf{A}); \mathbf{Y}) = I(\mathbf{X}; \mathbf{Y}) + I(\mathbf{A}; \mathbf{Y}|\mathbf{X}). \tag{8}$$

Above equations indicate f learns information from $I(\mathbf{X}; \mathbf{Y})$ and $I(\mathbf{A}; \mathbf{Y}|\mathbf{X})$ to predict labels. Since structural attacks are designed to mislead GNNs, the target of structural attacks is to seek an optimal modified structure $\hat{\mathbf{A}}$ to make GNNs unable to predict labels correctly. This target can be formulated as the minimizing of $I(f(\mathbf{X}, \hat{\mathbf{A}}); \mathbf{Y})$:

$$\delta^* = \arg\min_{\delta} I(f(\mathbf{X}, \mathbf{A} + \delta); \mathbf{Y}), \quad \hat{\mathbf{A}} = \mathbf{A} + \delta^*. \tag{9}$$

According to Eq. 8, this equation equal to minimize the sum of $I(\mathbf{X}; \mathbf{Y})$ and $I(\hat{\mathbf{A}}; \mathbf{Y}|\mathbf{X})$:

$$\delta^* = \arg\min_{\delta} I(\mathbf{X}; \mathbf{Y}) + I((\mathbf{A} + \delta); \mathbf{Y}|\mathbf{X}). \tag{10}$$

Since $I(\mathbf{X}; \mathbf{Y})$ is irrelevant to the structure $\mathbf{A}$ and δ, structural attack is actually minimizing the conditional mutual information $I((\mathbf{A} + \delta); \mathbf{Y}|\mathbf{X})$. We summarize the above analysis results as Lemma. 1.

Lemma 1 (Essence of Structural Attacks). *The essence of structural attacks is minimizing $I(\hat{\mathbf{A}}; \mathbf{Y}|\mathbf{X})$, which is the conditional mutual information between the modified structures and the labels under the given node attributes.*

For victim GNNs, the minimized $I(\hat{\mathbf{A}}; \mathbf{Y}|\mathbf{X})$ implies a reduction in the amount of useful information that can be learned, thereby degrading the performance of these GNNs.

To intuitively demonstrate Lemma. 1, we experimentally measure the value of $I(\hat{\mathbf{A}}; \mathbf{Y}|\mathbf{X})$. Directly computing $I(\hat{\mathbf{A}}; \mathbf{Y}|\mathbf{X})$ is unfeasible, as it requires all possible values of $\mathbf{X}$, which are unavailable in the dataset. Therefore, we propose an indirect method detailed in Appendix. F to calculate the approximate value MI_c.

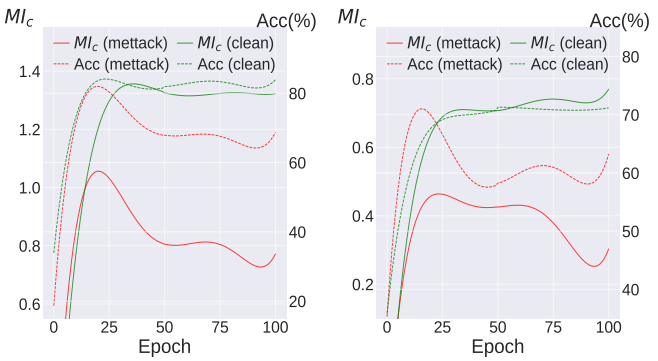


Fig. 2: Visualization of conditional mutual information and node classification accuracy on (a) Cora and (b) Citeseer.

As shown in Fig. 2, we conduct node classification experiments on Cora and Citeseer datasets. During training, we measure MI_c for GCN at each epoch and compare its performance when trained on a clean graph versus a graph modified by Mettack with a 20% perturbation ratio. From Fig. 2, we can observe that MI_c of the GCN trained on the modified structure (red solid lines) is significantly lower than that of the GCN trained on the clean graph (green solid lines). Moreover, the trend of MI_c (solid lines) is consistent with the trend of accuracy (dashed lines). When MI_c of the GCN trained on the modified structure decreases (at 10-50 epochs), the accuracy also drops significantly. This observation indicates that the essence of structural attack is to reduce the conditional mutual information to degrade the performance of GNNs, consistent with Lemma. 1. We provide more visualizations of MI_c for various GNN models on various datasets in Appendix. H to demonstrate the universality of Lemma. 1.

B. Pairwise Effect of Structural Attacks

The above analysis reveals the pairwise effect: to maximize impact, the modified structures must be paired with the original node attributes $\mathbf{X}$. If victim GNNs use modified structures with alternative attributes $\mathbf{X}^{'}$, the attack's effectiveness decreases. We present Theorem 1 as follows:

Theorem 1 (**Pairwise Effect of Structural Attacks**). *When the victim GNN f receives the modified structure $\hat{\mathbf{A}}$ paired with alternative node attributes $\mathbf{X}^{'}$ as inputs, the structural attack is less effective compared to when $\hat{\mathbf{A}}$ is paired with $\mathbf{X}$:*

$$I(\hat{\mathbf{A}};\mathbf{Y}|\mathbf{X}) \leq I(\hat{\mathbf{A}};\mathbf{Y}|\mathbf{X}^{'}), \tag{11}$$

$$I(f(\mathbf{X},\hat{\mathbf{A}});\mathbf{Y}) \leq I(f(\mathbf{X}^{'},\hat{\mathbf{A}});\mathbf{Y}). \tag{12}$$

We conduct experiments to intuitively demonstrate the paired effects of the case. We evaluate the node classification accuracy on the Cora dataset for a GCN model trained with different input pairs under the Mettack attack, using five attack ratios ranging from 0% (clean graph) to 20%. In pairs $(\mathbf{X}^{'}_{(0.01)},\hat{\mathbf{A}})$ and $(\mathbf{X}^{'}_{(0.05)},\hat{\mathbf{A}})$, we replace $\mathbf{X}$ with $\mathbf{X}^{'}_{\alpha}$, which is generated by adding Gaussian noise to $\mathbf{X}$, where α is the noise ratio. The results in Table. II indicate that $(\mathbf{X}^{'}_{(0.01)},\hat{\mathbf{A}})$ and $(\mathbf{X}^{'}_{(0.05)},\hat{\mathbf{A}})$ do not affect the performance of the GCN on a clean graph, as the noise ratios are tiny. However, they consistently show improvements in accuracy compared to the input pair $(\mathbf{X},\hat{\mathbf{A}})$. This demonstrates that $\hat{\mathbf{A}}$ paired with $\mathbf{X}^{'}$ diminishes the effectiveness of structural attacks.

TABLE II: The effects of different input pairs.

Pairs	$(\mathbf{X},\hat{\mathbf{A}})$	$(\mathbf{X}^{'}_{(0.01)},\hat{\mathbf{A}})$	$(\mathbf{X}^{'}_{(0.05)},\hat{\mathbf{A}})$
clean	83.0	83.6 (↑ **0.6**)	83.0 (0.0)
5%	76.7	77.7 (↑ **1.0**)	77.6 (↑ **0.9**)
10%	72.7	73.8 (↑ **1.1**)	73.9 (↑ **1.2**)
15%	68.6	70.9 (↑ **2.3**)	70.4 (↑ **1.8**)
20%	64.2	65.7 (↑ **1.5**)	65.7 (↑ **1.5**)

C. Defense by Disrupting Pairwise Effect

Theorem. 1 and Table. II suggest a promising strategy to enhance the robustness of GNNs against structural attacks: **disrupting pairwise effect**, i.e., mismatching $\mathbf{X}$ and $\hat{\mathbf{A}}$. The main idea of existing robust GNNs falls within the scope of disrupting the pairwise effect, which focuses on purifying modified structures and altering the modified structure $\hat{\mathbf{A}}$ with a purified structure $\hat{\mathbf{A}}^{'}$ to form a new input pair ($\mathbf{X}$, $\hat{\mathbf{A}}^{'}$) for models, thus disrupting the pair. However, purifying modified structures requires defenders to identify and remove the perturbed edges correctly, which is challenging due to the lack of prior information and is always time-consuming.

In contrast to existing efforts, we propose a more efficient method of disrupting the pair: altering node attributes $\mathbf{X}$ with $\mathbf{X}^{'}$ to render structural attacks inefficient. Generating a proper $\mathbf{X}^{'}$ is a critical issue, since $\mathbf{X}^{'}$ should improve the models' robustness while ensuring effective training. Therefore, we outline conditions that a proper $\mathbf{X}^{'}$ should meet in Theorem. 2.

Theorem 2 (**Conditions of Alternative Node Attributes**). *To enable GNNs to defend against structure attacks while ensuring normal training, $\mathbf{X}^{'}$ should satisfy the following conditions:*

$$(1)\quad I(\mathbf{X}^{'};\mathbf{Y}) \geq I(\mathbf{X};\mathbf{Y}), \tag{13}$$

$$(2)\quad I(\hat{\mathbf{A}};\mathbf{Y}|\mathbf{X}^{'}) > I(\hat{\mathbf{A}};\mathbf{Y}|\mathbf{X}). \tag{14}$$

The first condition in Theorem. 2 requires the alternative node attributes to share equal or more mutual information with labels than the original node attributes, avoiding potential information loss caused by $\mathbf{X}^{'}$. The second condition is based on Theorem 1, which guarantees that the conditional mutual information under $\mathbf{X}^{'}$ is more than that under $\mathbf{X}$, thus diminishing the effectiveness of structural attacks.

According to Theorem. 2, encouraging GNN models to learn information from $I(\mathbf{A};\mathbf{Y}|\mathbf{X}')$ can enhance robustness, under the condition that $\mathbf{X}^{'}$ does not cause information loss. In the next section, we propose two simple training strategies to disrupt the pairwise effect while simultaneously meeting both conditions.

VI. Methodology

In this section, we present our methodology for improving the robustness of GNNs against structural attacks. The essence of our approach lies in disrupting the pairwise effect through two simple but effective training strategies.

A. Intuition and Overview

Based on our analysis in Section V, a promising approach to defend against structural attacks is disrupting the pairwise effect. A straightforward way is directly constructing a new node attributes $\mathbf{X}^{'}$ that satisfy the conditions specified in Theorem. 2. However, this is challenging because the dimension of $\mathbf{X}^{'}$ is $n \times d$, where both n and d are often very large, potentially thousands. Directly constructing an $\mathbf{X}^{'}$ of such dimensions is time-consuming and computationally intensive.

Instead, our solution is to *implicitly* disrupt the pairwise effect by modifying the trained node embeddings Z in a principled way. Importantly, this strategy can be efficiently implemented through re-designing the training process of GNNs, thus significantly reducing the computational overhead.

Specifically, we propose two training strategies: Structural Fine-tuning (SF) and Progressive Structural Training (PST). The main idea of these strategies is to utilize trained node embeddings Z to replace $\mathbf{X}$ and encourage GNN models to learn information from $I(\hat{\mathbf{A}}; \mathbf{Y}|Z)$, instead of $I(\hat{\mathbf{A}}; \mathbf{Y}|\mathbf{X})$, thereby disrupting the pairwise effect. The difference between these two training strategies lies in how to obtain the trained node embeddings Z: SF through node attributes pre-training and PST through progressive structure training. Since the training strategies for generating Z solely involve changes to the training process and do not require modifications to the GNN framework or its core components, they can be widely applied to various GNN models. In addition, these training strategies will not significantly increase the training time or the complexity of hyperparameter tuning.

B. Progressive Structural Training (PST)

Implementation. PST divides the entire training process of GNNs into T training stages. The $t-$th training stage employees the pair $(\mathbf{X}, \hat{\mathbf{A}}_t)$ to train the model, where $\hat{\mathbf{A}}_t$ is a edges subset sampled from $\hat{\mathbf{A}}$ at a sampling ratio $\alpha_t \in [0, 1]$. In our implementation, we set α_t as $t-$th T-quantile. As training progresses, the sampling ratio α_t is gradually increased until the final stage, where the entire $\hat{\mathbf{A}}$ is used. Once the training in the current stage converges, indicated by the training loss no longer decreasing, the process progresses to the next stage. In each training stage, the model is initialized with the parameters learned from the previous stage and is further trained on a subset of modified structures with an increased sampling ratio and node attributes, building upon the previous training stage. We show PST in Algo. 1.

Theoretical Analysis. The main idea of PST is leveraging the node embeddings obtained from the previous stage as alternative node attributes in each training stage, thereby disrupting the pairwise effect and satisfying the conditions outlined in Theorem. 2.

Algorithm 1 Progressive Subset Training (PST)

Require: Input graph $\mathcal{G}'(\mathbf{X}, \hat{\mathbf{A}}, \mathbf{Y})$, a GNN model f with parameters Θ, number of training stages T, sampling ratio $\{\alpha_t | t = 1, 2, ..., T\}$
Ensure: Trained model f_Θ
1: Initialize f with random parameter Θ
2: **for** $t = 1$ to T **do**
3: $\quad \hat{\mathbf{A}}_t \leftarrow$ random sampling edges from $\hat{\mathbf{A}}$ with α_t
4: $\quad \Delta\Theta^* = \arg\min \mathcal{L}(f_{\Theta+\Delta\Theta}(\hat{\mathbf{A}}_t, \mathbf{X}), \mathbf{Y})$
5: $\quad \Theta = \Theta + \overset{\Delta\Theta}{\Delta\Theta}^*$
6: **end for**
7: **return** f_Θ

For ease of understanding the main idea of PST, we suppose the GNN model f as a 1-layer SGC model [40] without the activation function. The SGC is chosen because of its simple structure, which linearizes the GNN model and eliminates unnecessary redundant computations and additional complexity.

According to the aggregation scheme of SGC, the node embeddings Z_t of the $t-$th training stage is:

$$Z_t = f_{\Theta_t}(\mathbf{X}, \hat{\mathbf{A}}_t) = \hat{\mathbf{A}}_t \mathbf{X} \Theta_t, \quad \Theta_t = \Theta_{t-1} + \Delta\Theta, \tag{15}$$

where Θ_t is the parameters of SGC at $t-$th training stage. Since Z_t is continuously trained based on $(t-1)$-th training stage, the $t-$th training stage actually trains an offset $\Delta\Theta$ and applies it to Θ_{t-1} to obtain Θ_t. Thus we have:

$$Z_t = \hat{\mathbf{A}}_t \mathbf{X}(\Theta_{t-1} + \Delta\Theta) = \hat{\mathbf{A}}_t(\mathbf{X}\Theta_{t-1} + \mathbf{X}\Delta\Theta). \tag{16}$$

For $(t-1)$-th training stage, we have $Z_{t-1} = \hat{\mathbf{A}}_{t-1}\mathbf{X}\Theta_{t-1}$. We assume $\hat{\mathbf{A}}_{t-1}^{-1}$ holds. Although this may not always hold true in practical scenarios, we make this assumption to facilitate our theoretical analysis and proofs. We substitute $\hat{\mathbf{A}}_{t-1}^{-1} Z_{t-1} = \mathbf{X}\Theta_{t-1}$ into the above equation, and obtain:

$$Z_t = \hat{\mathbf{A}}_t(\hat{\mathbf{A}}_{t-1}^{-1} Z_{t-1} + \mathbf{X}\Delta\Theta) = \hat{\mathbf{A}}_t \cdot Z_{\Delta\Theta_t}, \tag{17}$$

where $Z_{\Delta\Theta_t} = (\hat{\mathbf{A}}_{t-1}^{-1} Z_{t-1} + \mathbf{X}\Delta\Theta)$ is a linear combination of Z_{t-1} and $\mathbf{X}$, parametered by $\Delta\Theta$. According to Eq. (7) and Eq. (8), the well-trained f_{Θ_t} fulfills:

$$I(f_{\Theta_t}(\mathbf{X}, \hat{\mathbf{A}}_t); \mathbf{Y}) = I(Z_t; \mathbf{Y}) = I(\hat{\mathbf{A}}_t \cdot Z_{\Delta\Theta_t}; \mathbf{Y}) \tag{18}$$

$$= I(Z_{\Delta\Theta_t}; \mathbf{Y}) + I(\hat{\mathbf{A}}_t; \mathbf{Y}|Z_{\Delta\Theta_t}). \tag{19}$$

It means that in the t-th training stage, the original node attributes are replaced with $Z_{\Delta\Theta_t}$ actually. The model learns information from two parts: $I(Z_{\Delta\Theta_t}; \mathbf{Y})$ and $I(\hat{\mathbf{A}}_t; \mathbf{Y}|Z_{\Delta\Theta_t})$. Each training stage trains the optimal $\Delta\Theta$ to regulate the proportions of $\mathbf{X}$ and Z_{t-1}, thereby obtaining the optimal $Z_{\Delta\Theta_t}$ for the current training stage.

Since $Z_{\Delta\Theta_t}$ contains rich information from Z_{t-1} and $\mathbf{X}$, it serves as a high-quality node embedding that satisfies the conditions outlined in Theorem. 2. We provide the following theorem and its proofs in Appendix. D.

Theorem 3. *For the t-th training stage, PST pursues the model to learn information from $I(Z_{\Delta\Theta_t};\mathbf{Y})$ and $I(\hat{\mathbf{A}}_t;\mathbf{Y}\mid Z_{\Delta\Theta_t})$, which fulfill the conditions:*

$$I(Z_{\Delta\Theta_t};\mathbf{Y}) \geq I(\mathbf{X};\mathbf{Y}),$$
$$I(\hat{\mathbf{A}}_t;\mathbf{Y}\mid Z_{\Delta\Theta_t}) > I(\hat{\mathbf{A}};\mathbf{Y}\mid\mathbf{X}).$$

The final training stage utilizes the node embeddings from the previous training stages along with all the structural information ($\hat{\mathbf{A}}_T = \hat{\mathbf{A}}$) to generate node embeddings for downstream tasks. At this point, the node embeddings encompass information from all prior training stages as well as all the structural information.

Differences between PST and Edge Removing. While the structure subset sampling in PST shares similarities with robust GNNs that employ edge-removing or edge-dropping [16], [41], [42], it is important to note that their objectives differ significantly. PST leverages subsets of the structure to generate high-quality node embeddings, whereas edge-removing focuses on removing malicious edges injected by attackers. The latter is often time-consuming, as accurately identifying malicious edges is inherently challenging. Additionally, removing benign edges may lead to the loss of structural information and degrade model performance.

C. Structural Fine-tuning (SF)

SF initially pre-trains GNNs using only the node attributes to generate node embeddings Z_P, which captures rich attribute information. Subsequently, SF fine-tunes the model with a modified adjacency matrix to integrate structural information. In this process, the original node attributes are replaced by a linear combination of itself and the pre-trained node embeddings Z_P, thus disrupting the pairwise effect. Since Z_P is derived from node attributes, it satisfies the first condition in Theorem 2, indicating that it shares as much mutual information with the labels as the original node attributes. During fine-tuning, we introduce inter-class augmentation and employ graph contrastive learning techniques to enhance the model's ability to learn structural information.

a) Attributes Pre-training Stage: Attributes pre-training is employed to learn node embeddings solely from node attributes $\mathbf{X}$. Specifically, a randomly initialized GNN model f_Θ with parameters Θ takes node attributes $\mathbf{X}$ of the input graph and an identity matrix $\mathbf{E}$ as inputs and aims to minimize $\mathcal{L}$ to learn node embeddings Z_P:

$$\Theta_p = \arg\min_{\Theta} \mathcal{L}(f_\Theta(\mathbf{X},\mathbf{E}),\mathbf{Y}), \quad Z_P = f_{\Theta_p}(\mathbf{X},\mathbf{E}). \quad (20)$$

Lines 3-7 of Alg. 2 show the attributes pre-training stage. We set the pre-training epochs to be identical to the training epochs of the vanilla GNNs in our implementation.

The well-trained model f_{Θ_p} should learn sufficient information from $\mathbf{X}$ to predict labels $\mathbf{Y}$ and thus guarantee that the embedding Z_P shares as much mutual information with the labels as the original node attributes. However, the lack of structural information makes Z_P insufficient to predict labels accurately. Hence, we propose structure fine-tuning, which adjusts Z_P using $\hat{\mathbf{A}}$ to incorporate structural information.

Algorithm 2 Structural Fine-tuning (SF)

1: **Input**: Input graph $\mathcal{G}'(\mathbf{X},\hat{\mathbf{A}},\mathbf{Y})$, identity matrix $\mathbf{E}$ and GNN model f with parameter θ, pre-training epoch $pre-train\_epoch$, fine-tuning epoch $finetune\_epoch$, fine-tuning loss $\mathcal{L}_f$ and contrastive loss $\mathcal{L}_c$, node set V
2: */*Attributes pre-training*/*
3: **for** $e = 1, 2, ..., pre-train\_epoch$ **do**
4: $\quad Z_\theta = f_\theta(\mathbf{X},\mathbf{E})$
5: $\quad \theta = \theta + \nabla_\theta\ \mathcal{L}(Z_\theta,\mathbf{Y})$
6: **end for**
7: θ_p = θ
8: */*Inter-class Node Attributes Augmentation*/*
9: empty set $\mathbf{X}_{\mathbf{inter}}$.
10: **for** $v \in V$ **do**
11: $\quad c = v.class$
12: $\quad$ InterClassSet=$\{V.class \neq c\}$
13: $\quad$ num = v.degree
14: $\quad inter\_class$ =RandomChoice(num,InterClassSet)
15: $\quad \mathbf{X}_{v\_inter} = mean(\mathbf{X}_{inter\_class})$
16: $\quad$ add $\mathbf{X}_{v\_inter}$ into $\mathbf{X}_{\mathbf{inter}}$
17: **end for**
18: */*Structure fine-tuning*/*
19: **for** $e = 1, 2, ..., finetune\_epoch$ **do**
20: $\quad Z_F = f_{\theta_p}(\mathbf{X},\hat{\mathbf{A}}), \quad Z_{inter} = f_{\theta_p}(\mathbf{X}_{\mathbf{inter}},\hat{\mathbf{A}})$
21: $\quad \mathcal{L}_f = \mathcal{L}(Z_F,\mathbf{Y}) + \mathcal{L}_c(Z_F, Z_{inter})$
22: $\quad \theta_p = \theta_p + \nabla_{\theta_p}\mathcal{L}_f$
23: **end for**
24: $\theta^* = \theta_p$
25: **return** f_{θ^*}

b) Structure Fine-tuning Stage: The target of the structure fine-tuning stage is to incorporate as much structural information as possible into Z_P so that it can be used to predict labels accurately. Therefore, the model is initialized by the pre-trained parameters Θ_p and trained to minimize $\mathcal{L}$:

$$\Theta^* = \arg\min_{\Delta\Theta} \mathcal{L}(Z_F,\mathbf{Y}), \quad Z_F = f_{\Theta_p+\Delta\Theta}(\mathbf{X},\hat{\mathbf{A}}). \quad (21)$$

By fine-tuning the model initialized with the pre-trained parameters Θ_p, SF implicitly replaces the original node attributes to disrupt the pairwise effect. Similar to the analysis of PST, we suppose f as a 1-layer SGC model and have:

$$Z_P = \mathbf{E}\mathbf{X}\Theta_P = \mathbf{X}\Theta_P, \quad Z_F = \hat{\mathbf{A}}\mathbf{X}(\Theta_P + \Delta\Theta). \quad (22)$$

Z_F can be represented as:

$$Z_F = \hat{\mathbf{A}}(Z_P + \mathbf{X}\Delta\Theta) = \hat{\mathbf{A}}\cdot Z_{\Delta\Theta}, \quad (23)$$

where $Z_{\Delta\Theta}$ is the linear combination of Z_P and $\mathbf{X}$. According to Eq. (7) and Eq. (8), the well-trained f_{Θ^*} fulfills:

$$I(f_{\Theta^*}(\mathbf{X},\hat{\mathbf{A}});\mathbf{Y}) = I(Z_F;\mathbf{Y}) = I(\hat{\mathbf{A}}\cdot Z_{\Delta\Theta};\mathbf{Y}) \quad (24)$$
$$= I(Z_{\Delta\Theta};\mathbf{Y}) + I(\hat{\mathbf{A}};\mathbf{Y}|Z_{\Delta\Theta}), \quad (25)$$

which indicates SF replaces original node attributes with $Z_{\Delta\Theta}$. It means SF pursues the model to learn information from $I(Z_{\Delta\Theta};\mathbf{Y})$ and $I(\hat{\mathbf{A}};\mathbf{Y}|Z_{\Delta\Theta})$, instead of $I(\mathbf{X};\mathbf{Y})$ and $I(\hat{\mathbf{A}};\mathbf{Y}|\mathbf{X})$, thus disrupting the pairwise effect. Theorem 4 demonstrates that $Z_{\Delta\Theta}$ fulfills the conditions in Theorem 2.

Theorem 4. *SF pursues the model to learn information from $I(Z_{\Delta\Theta};\mathbf{Y})$ and $I(\hat{\mathbf{A}};\mathbf{Y}\mid Z_{\Delta\Theta})$, which fulfill the conditions:*

$$I(Z_{\Delta\Theta};\mathbf{Y}) \geq I(\mathbf{X};\mathbf{Y}),$$
$$I(\hat{\mathbf{A}};\mathbf{Y}\mid Z_{\Delta\Theta}) > I(\hat{\mathbf{A}};\mathbf{Y}\mid\mathbf{X}).$$

We provide proofs in Appendix. E.

To further enhance the structural information that the model can learn during the fine-tuning, we introduce contrastive learning techniques and propose Inter-class Node Attributes Augmentation (InterNAA). Specifically, InterNAA generates a view $\mathcal{G}_{inter} = (\mathbf{X}_{inter}, \hat{\mathbf{A}}, \mathbf{Y})$ of the input graph by replacing the node attributes of each node v in the training set with the average node attributes of several nodes with different class as v that are randomly sampled from the training set. The number of samples equals the degree of the node v. The process of InterNAA is shown in Lines 9-17 of Alg. 2.

Since the node attributes of $\mathbf{X}_{inter}$ are replaced by attributes from nodes belonging to different classes, the pre-trained model f cannot extract useful information from $\mathbf{X}_{inter}$ to predict labels accurately. Therefore, f primarily extracts structural information from $\mathcal{G}_{inter}$, and Z_{inter} contains most of the structural information. By aligning Z_F to Z_{inter} during fine-tuning, SF encourages Z_F to learn more structural information.

In summary, during fine-tuning, the model f takes both the input graph $\mathcal{G}^{'}$ and $\mathcal{G}_{inter}$ as inputs to produce node embeddings Z_F and Z_{inter}, respectively, while minimizing $\mathcal{L}$ and the contrastive loss function $\mathcal{L}_c$:

$$\theta^* = \arg\min_{\Delta\theta} \mathcal{L}(Z_F, \mathbf{Y}) + \mathcal{L}_c(Z_F, Z_{inter}),$$
$$Z_F = f_{\theta_p+\Delta\theta}(\mathbf{X_{train}}, \hat{\mathbf{A}}), \quad Z_{inter} = f_{\theta_p+\Delta\theta}(\mathbf{X_{inter}}, \hat{\mathbf{A}}), \tag{26}$$

where $\mathcal{L}_c$ is any typical contrastive function such as InfoNCE [43]. We show the entire SF in Alg. 2. The only hyperparameter of SF is the finetuning epoch, typically set between 3 and 5. The procedure for selecting the optimal number of epochs is detailed in Sec. VII.

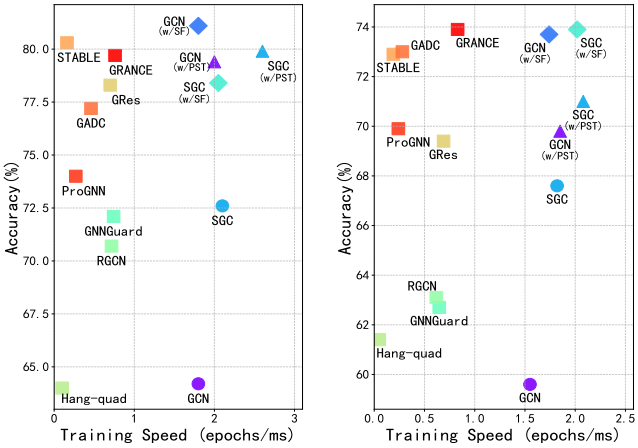
D. Comparison between PST and SF

Both PST and SF improve the robustness of GNNs by disrupting pairwise effects. However, they differ significantly in two aspects:

1. Structural Integrity: PST learns node embeddings from structural subsets, potentially overlooking multi-hop neighbors and resulting in the loss of structural integrity and multi-hop information. In contrast, SF leverages the entire adjacency matrix during fine-tuning, thereby preserving structural integrity. Consequently, SF generally outperforms PST, especially in GNN models (e.g., EvenNet) that rely on multi-hop neighborhood information (more analysis in Sec. VII-B).

2. Training Process: PST employs a simple end-to-end training strategy, whereas SF adopts a two-stage approach involving pre-training and fine-tuning. As a result, PST is faster and more convenient to deploy (as shown in Sec. VII).

The above analysis indicates that each strategy has its advantages and disadvantages. We propose two strategies to enable defenders to choose the most suitable one for their application scenarios. Specifically, PST enables faster training, is suitable for time-sensitive scenarios such as fraud detection in online recommendation systems [9], [10]. In contrast, SF preserves structural integrity, making it a recommended choice for applications that rely on structural integrity (e.g., bioinformatics [3]).



(a) Cora ptb=20% (b) Citeseer ptb=20%

Fig. 3: Comparison of robustness and training speed.

VII. Experiments

We conduct experiments on the node classification task to validate the effectiveness of the proposed training strategies and aim to address the following research questions (RQs):

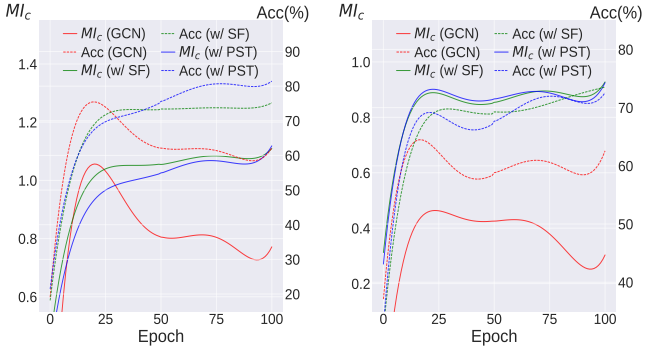
RQ1: Are our proposed training strategies effective in defending against structural attacks?

RQ2: Can the proposed training strategies be applied to various GNN variants, and can they further enhance the robustness of robust GNNs?

RQ3: How efficient are the proposed training strategies, and are they able to improve the training speed of GNNs?

RQ4: Can the proposed training strategies effectively disrupt the pairwise effect?

RQ5: How do the proposed training strategies perform in terms of robustness and efficiency on large-scale and heterophilic graphs?



(a) Cora ptb=20% (b) Citeseer ptb=20%

Fig. 4: Comparison of MI_c and accuracy on Cora and Citeseer under Mettack with 20% perturbation ratio.

A. Datasets, Baselines and Implementation

We evaluate our approach on widely used benchmarks: Cora [44], CiteSeer [45], and Pubmed [46], as well as large-

TABLE III: Average classification accuracy (± standard deviation) of 10 runs under **Mettack** with different perturbation ratios (ptb). The best and second-best results are highlighted in bold and underlined, respectively.

	Dataset	Cora			Citeseer			Pubmed		
	ptb(%)	Clean	10%	20%	Clean	10%	20%	Clean	10%	20%
Vanilla GNNs	GCN	82.9±0.3	72.7±0.7	64.2±0.1	71.7±0.3	65.2±0.3	59.6±0.6	86.9±0.4	82.0±0.0	80.1±0.1
	w/ PST (Improv.)	82.5±0.5 (↓ 0.4)	81.0±0.7 (↑ 8.4)	81.1±0.8 (↑ 16.9)	73.4±0.5 (↑ 1.7)	72.7±0.8 (↑ 7.5)	69.8±0.9 (↑ 10.2)	86.2±0.5 (↓ 0.7)	85.0±0.2 (↑ 3.0)	84.8±0.3 (↑ 4.7)
	w/ SF (Improv.)	83.4±0.5 (↑ 0.5)	82.1±0.6 (↑ 9.4)	79.4±0.7 (↑ 15.2)	75.1±0.4 (↑ 3.4)	74.3±0.4 (↑ 9.1)	73.7±0.4 (↑ 14.1)	85.4±0.4 (↓ 1.5)	85.1±0.3 (↑ 3.1)	84.5±0.4 (↑ 4.4)
	SGC	83.1±0.5	76.3±0.6	72.6±0.8	71.9±0.4	70.3±0.7	67.6±0.5	86.3±0.1	82.5±0.1	79.3±0.0
	w/ PST (Improv.)	82.6±0.6 (↓ 0.5)	81.5±0.7 (↑ 5.2)	79.9±0.7 (↑ 7.3)	73.4±0.4 (↑ 1.5)	73.0±0.7 (↑ 2.7)	71.0±0.8 (↑ 3.4)	86.4±0.0 (↑ 0.1)	85.1±0.0 (↑ 2.6)	84.7±0.1 (↑ 5.4)
	w/ SF (Improv.)	83.0±0.1 (↓0.1)	80.3±0.1 (↑ 4.0)	78.4±0.3 (↑ 5.8)	75.4±0.6 (↑ 3.5)	74.2±0.6 (↑ 3.9)	73.9±0.7 (↑ 6.3)	86.4±0.0 (↑ 0.1)	85.5±0.1 (↑ 3.0)	85.0±0.1 (↑ 5.7)
	GAT	82.0±0.4	80.1±0.7	79.5±0.6	71.9±0.4	68.7±0.6	61.0±0.8	86.5±0.1	82.3±0.1	78.6±0.2
	w/ PST (Improv.)	82.3±0.8 (↑ 0.3)	81.0±0.9 (↑ 0.9)	79.5±0.6 (0.0)	73.4±0.4 (↑ 1.5)	72.0±0.5 (↑ 3.3)	71.0±0.7 (↑ 10.0)	86.4±0.0 (↓0.1)	85.5±0.2 (↑ 3.2)	84.8±0.1 (↑ 6.2)
	w/ SF (Improv.)	82.8±0.9 (↑ 0.8)	**82.2±0.9** (↑ 2.1)	80.6±0.5 (↑ 1.1)	75.1±0.4 (↑ 3.2)	74.4±0.4 (↑ 5.7)	**74.3±0.2** (↑ 13.3)	86.5±0.1 (0.0)	85.8±0.4 (↑ 3.5)	85.0±0.3 (↑ 6.4)
	APPNP	83.2±0.3	69.3±0.8	61.3±0.6	71.4±0.2	62.2±0.5	53.4±0.6	86.1±0.2	81.1±0.1	78.4±0.3
	w/ PST (Improv.)	83.1±0.5 (↓0.1)	81.4±0.4 (↑ 12.1)	80.1±0.5 (↑ 18.8)	73.3±0.7 (↑ 1.9)	72.6±0.4 (↑ 10.4)	71.0±0.7 (↑ 17.6)	85.9±0.2 (↓0.2)	83.6±0.3 (↑ 2.5)	81.1±0.2 (↑ 2.7)
	w/ SF (Improv.)	83.4±0.1 (↑ 0.2)	79.8±0.3 (↑ 10.5)	79.1±0.2 (↑ 17.8)	75.0±0.6 (↑ 3.6)	74.1±0.1 (↑ 11.9)	73.5±0.5 (↑ 20.1)	86.7±0.3 (↑ 0.6)	84.0±0.1 (↑ 2.9)	82.5±0.3 (↑ 4.1)
	GCNII	80.3±0.3	65.8±0.4	56.6±0.8	71.9±0.4	64.1±0.6	57.5±0.6	85.8±0.1	79.5±0.3	76.2±0.4
	w/ PST (Improv.)	78.3±0.1 (↓2.0)	75.2±0.5 (↑ 9.4)	72.8±0.6 (↑ 16.2)	72.0±0.6 (↑ 0.1)	69.9±0.7 (↑ 5.8)	67.1±0.4 (↑ 9.6)	86.3±0.1 (↑ 0.5)	83.4±0.2 (↑ 3.9)	80.6±0.3 (↑ 4.4)
	w/ SF (Improv.)	80.3±0.3 (0.0)	77.0±0.8 (↑ 11.2)	75.4±0.1 (↑ 18.8)	73.0±0.1 (↑ 1.1)	72.7±0.4 (↑ 8.6)	70.9±0.1 (↑ 13.4)	86.2±0.2 (↑ 0.4)	83.9±0.3 (↑ 4.4)	81.7±0.2 (↑ 5.5)
Robust GNNs	RGCN	82.7±0.8	74.4±0.7	70.7±0.6	71.7±0.4	65.3±0.5	63.1±0.7	86.6±0.2	78.4±0.3	72.3±0.1
	w/ PST (Improv.)	82.4±0.7 (↓0.3)	80.1±0.5 (↑ 5.7)	75.8±0.4 (↑ 5.1)	73.4±0.5 (↑ 1.7)	72.0±0.5 (↑ 6.7)	70.7±0.6 (↑ 7.6)	86.5±0.3 (↓0.1)	84.2±0.3 (↑ 5.8)	81.1±0.2 (↑ 8.8)
	w/ SF (Improv.)	82.7±0.6 (0.0)	80.6±0.7 (↑ 6.2)	76.4±0.4 (↑ 5.7)	75.2±0.6 (↑ 3.5)	74.1±0.3 (↑ 8.8)	73.0±0.6 (↑ 9.9)	86.8±0.1 (↑0.2)	85.6±0.5 (↑ 7.2)	82.0±0.3 (↑ 9.7)
	GNNGuard	81.8±0.5	77.2±0.4	72.1±0.5	72.0±0.7	66.8±0.4	62.7±0.8	87.1±0.1	80.1±0.2	77.8±0.3
	w/ PST (Improv.)	82.1±0.3 (↑0.3)	80.3±0.4 (↑ 3.1)	78.8±0.4 (↑ 6.7)	73.1±0.5 (↑ 1.1)	73.0±0.8 (↑ 6.2)	70.6±0.9 (↑ 7.9)	87.1±0.2 (0.0)	85.4±0.3 (↑ 5.3)	82.1±0.2 (↑ 4.3)
	w/ SF (Improv.)	83.0±0.4 (↑1.2)	80.9±0.5 (↑ 3.7)	79.0±0.5 (↑ 6.9)	75.1±0.7 (↑ 3.1)	73.7±0.6 (↑ 6.9)	73.3±0.9 (↑10.6)	**87.2±0.1** (↑0.1)	86.1±0.2 (↑ 6.0)	81.7±0.3 (↑ 3.9)
	GCN-Jaccard	82.3±0.7	76.9±1.2	75.2±1.7	71.4±0.8	67.7±1.1	66.5±0.6	86.2±0.1	79.6±0.2	70.5±0.4
	w/ PST (Improv.)	82.5±0.6 (↑ 0.2)	81.2±0.5 (↑ 4.3)	80.0±0.5 (↑ 4.8)	73.3±0.5 (↑ 1.9)	72.8±0.8 (↑ 5.1)	70.9±0.7 (↑ 4.4)	86.5±0.2 (↑ 0.3)	84.5±0.0 (↑ 4.9)	80.9±0.2 (↑ 10.4)
	w/ SF (Improv.)	82.2±0.9 (↓0.1)	81.9±0.8 (↑ 5.0)	**81.2±0.9** (↑ 6.0)	75.2±0.4 (↑ 3.8)	**74.7±0.4** (↑ 7.0)	73.9±0.4 (↑ 7.4)	86.8±0.1 (↑ 0.6)	85.0±0.2 (↑ 5.4)	81.2±0.1 (↑ 10.7)
	SimP-GCN	82.1±0.6	79.0±0.9	76.1±2.0	73.8±0.7	72.1±0.7	70.9±0.5	87.1±0.1	86.0±0.2	85.7±0.2
	w/ PST (Improv.)	81.8±0.7 (↓0.3)	79.9±0.4 (↑ 0.9)	76.6±0.5 (↑ 0.5)	73.8±0.9 (0.0)	73.4±0.7 (↑ 1.3)	73.0±0.4 (↑ 2.1)	87.0±0.3 (↓0.1)	86.5±0.2 (↑ 0.5)	85.4±0.0 (↓0.3)
	w/ SF (Improv.)	82.3±0.6 (↑ 0.2)	80.8±0.5 (↑ 1.8)	79.3±0.4 (↑ 3.2)	**75.5±0.4** (↑ 1.7)	74.6±0.5 (↑ 2.5)	73.3±0.9 (↑ 2.4)	87.1±0.2 (0.0)	**86.8±0.1** (↑ 0.8)	**85.9±0.2** (↑ 0.2)
	β-GNN	81.4±0.7	78.0±0.8	76.1±0.7	71.2±0.9	66.0±0.8	62.4±0.8	86.8±0.1	79.0±0.2	76.5±0.3
	w/ PST (Improv.)	82.0±0.5 (↑0.6)	79.7±0.3 (↑ 1.7)	77.5±0.8 (↑ 1.4)	72.8±0.8 (↑ 1.6)	73.1±0.7 (↑ 7.1)	69.9±0.7 (↑ 7.5)	87.0±0.1 (↑0.2)	86.1±0.4 (↑ 7.1)	80.0±0.2 (↑ 3.5)
	w/ SF (Improv.)	81.8±0.8 (↑0.4)	80.1±0.5 (↑ 2.1)	77.9±0.7 (↑ 1.8)	74.7±0.9 (↑ 3.5)	74.2±0.5 (↑ 8.2)	72.1±0.7 (↑ 9.7)	87.1±0.2 (↑0.3)	86.3±0.5 (↑ 7.3)	80.8±0.1 (↑ 4.3)
	NoisyGCN	82.9±0.6	75.6±1.2	74.5±1.3	72.3±0.4	71.2±0.4	69.8±0.9	85.0±0.0	67.4±0.2	56.5±0.4
	w/ PST (Improv.)	82.3±0.6 (↓0.6)	80.9±0.4 (↑ 5.3)	79.1±0.5 (↑ 4.6)	73.2±0.5 (↑ 0.9)	72.5±0.5 (↑ 1.3)	70.7±0.6 (↑ 0.9)	84.5±0.1 (↓0.5)	80.0±0.0 (↑ 12.6)	77.8±0.3 (↑ 21.3)
	w/ SF (Improv.)	83.2±0.9 (↑ 0.3)	**82.2±0.5** (↑ 6.6)	81.1±0.7 (↑ 6.6)	75.1±0.4 (↑ 2.8)	74.3±0.6 (↑ 3.1)	73.6±0.4 (↑ 3.8)	85.4±0.2 (↑ 0.4)	81.3±0.1 (↑ 13.9)	79.5±0.3 (↑ 23.0)
	EvenNet	83.1±0.4	77.8±1.1	78.2±0.9	73.8±0.5	73.3±0.4	73.2±0.5	86.7±0.1	85.6±0.2	85.3±0.2
	w/ PST (Improv.)	**83.6±0.5** (↑ 0.5)	80.2±0.7 (↑ 2.4)	78.0±0.6 (↓0.2)	71.3±0.7 (↓2.5)	71.0±0.5 (↓2.3)	70.6±1.0 (↓2.6)	86.5±0.2 (↓0.2)	85.5±0.1 (↓0.1)	84.8±0.1 (↓0.5)
	w/ SF (Improv.)	83.5±0.6 (↑ 0.4)	81.1±0.5 (↑ 3.3)	80.0±0.5 (↑ 1.8)	73.9±0.5 (↑ 0.1)	73.4±0.6 (↑ 0.1)	73.0±0.6 (↓0.2)	**87.2±0.3** (↑ 0.5)	86.1±0.2 (↑ 0.5)	85.4±0.1 (↑ 0.1)

scale graph datasets from [17]: ogbn-arxiv and ogbn-products. Additionally, we provide experimental results on heterophilic graphs [47]. For dataset details, please refer to Appendix. G.

To demonstrate that the proposed training strategies can be applied to various GNN variants, we conduct experiments on widely used vanilla GNNs—including GCN [15], GAT [25], GCNII [48], and APPNP [49]—as well as state-of-the-art robust GNNs such as RGCN [50], GNNGuard [34], $\beta-$GNN [51], GRes [52], GRANCE [36], GCN-Jaccard [12], SimP-GCN [13], EvenNet [38], and NoisyGCN [37]. However, it is important to note that several baselines cannot adopt our training strategies due to either not utilizing GNNs as their backbone (e.g., GADC [53]) or requiring specialized training procedures (e.g., STABLE [16], ProGNN [11]). Detailed discussion and configurations are deferred to the Appendix. G.

We select two representative structure attack methods: Mettack [7] and GraD [8], to verify the robustness of the proposed method and baselines. Source code and configuration of baselines are obtained from either the public implementation of DeepRobust [54], or the official implementation.

B. Defense Performance

We applied the proposed training strategies to various vanilla GNNs, including GCN, SGC, GAT, APPNP, and GC-

TABLE IV: Average classification accuracy (± standard deviation) of 10 runs under **GraD** attack with different perturbation ratios (ptb). The best and second-best results are highlighted in bold and underlined, respectively.

	Dataset	Cora			Citeseer			Pubmed		
	ptb(%)	5%	10%	20%	5%	10%	20%	5%	10%	20%
Vanilla GNNs	GCN	81.5±0.5	76.8±0.4	73.2±0.6	72.0±0.4	70.6±0.3	67.8±0.6	85.4±0.2	83.7±0.3	80.1±0.2
	w/ PST (Improv.)	82.2±0.4 (↑ 0.7)	80.6±0.5 (↑ 3.8)	77.4±0.8 (↑ 4.2)	74.6±0.5 (↑ 2.6)	73.8±0.4 (↑ 3.2)	71.2±0.5 (↑ 3.4)	86.3±0.1 (↑ 0.9)	85.6±0.1 (↑ 1.9)	84.4±0.2 (↑ 4.3)
	w/ SF (Improv.)	82.0±0.6 (↑ 0.5)	81.1±0.8 (↑ 4.3)	79.6±0.9 (↑ 6.4)	75.0±0.6 (↑ 3.0)	74.5±0.6 (↑ 3.9)	73.4±0.6 (↑ 5.6)	85.2±0.3 (↓ 0.2)	84.5±0.5 (↑ 0.8)	83.4±0.4 (↑ 3.3)
	SGC	82.1±0.4	80.0±0.6	75.2±0.6	70.6±0.4	66.8±0.4	62.5±0.6	86.0±0.1	83.4±0.1	81.5±0.1
	w/ PST (Improv.)	82.1±0.6 (0.0)	81.5±0.4 (↑ 1.5)	80.2±0.6 (↑ 5.0)	72.5±0.2 (↑ 1.9)	71.7±0.4 (↑ 4.9)	69.8±0.5 (↑ 7.3)	85.8±0.2 (↓ 0.2)	83.9±0.2 (↑ 0.5)	83.4±0.1 (↑ 1.9)
	w/ SF (Improv.)	82.4±0.8 (↑ 0.3)	81.1±0.7 (↑ 1.1)	80.0±0.4 (↑ 4.8)	74.8±0.3 (↑ 4.2)	73.4±0.6 (↑ 6.6)	72.2±0.6 (↑ 9.7)	85.9±0.1 (↓ 0.1)	84.6±0.3 (↑ 1.2)	83.8±0.2 (↑ 2.3)
	GAT	80.8±0.6	78.0±0.7	74.4±0.6	71.2±0.3	65.7±0.2	58.6±0.5	85.1±0.2	83.9±0.0	82.4±0.2
	w/ PST (Improv.)	81.2±0.4 (↑ 0.4)	80.0±0.5 (↑ 2.0)	78.9±0.7 (↑ 4.5)	72.6±0.7 (↑ 1.4)	71.4±0.5 (↑ 5.7)	70.8±0.4 (↑ 12.2)	86.0±0.0 (↑ 0.9)	85.6±0.1 (↑ 1.7)	85.1±0.2 (↑ 2.7)
	w/ SF (Improv.)	82.4±0.7 (↑ 1.6)	80.5±0.6 (↑ 2.5)	78.7±0.6 (↑ 4.3)	73.1±0.5 (↑ 1.9)	72.5±0.7 (↑ 6.8)	71.2±0.6 (↑ 12.6)	85.6±0.1 (↑ 0.5)	84.8±0.2 (↑ 0.9)	84.3±0.1 (↑ 1.9)
	APPNP	80.5±0.2	73.8±0.5	64.2±0.7	70.5±0.3	65.7±0.5	61.3±0.7	85.4±0.1	82.0±0.1	78.6±0.0
	w/ PST (Improv.)	82.0±0.6 (↑ 1.5)	80.9±0.3 (↑ 7.1)	76.4±0.5 (↑ 12.2)	73.2±0.6 (↑ 2.7)	71.4±0.4 (↑ 5.7)	70.5±0.4 (↑ 9.2)	85.3±0.2 (↓ 0.1)	82.2±0.2 (↑ 0.2)	79.5±0.1 (↑ 0.9)
	w/ SF (Improv.)	82.6±0.5 (↑ 2.1)	80.5±0.4 (↑ 6.7)	78.7±0.6 (↑ 14.5)	74.1±0.7 (↑ 3.6)	72.8±0.7 (↑ 7.1)	71.2±0.6 (↑ 9.9)	85.6±0.1 (↑ 0.2)	82.9±0.3 (↑ 0.9)	80.1±0.2 (↑ 1.5)
	GCNII	79.4±0.2	70.5±0.4	68.3±0.5	70.4±0.7	67.2±0.6	62.2±0.8	82.8±0.2	80.1±0.1	75.9±0.2
	w/ PST (Improv.)	79.0±0.5 (↓ 0.4)	73.4±0.5 (↑ 2.9)	71.1±0.4 (↑ 2.8)	70.9±0.7 (↑ 0.5)	69.3±0.9 (↑ 2.1)	68.0±0.8 (↑ 5.8)	85.7±0.1 (↑ 2.9)	82.2±0.1 (↑ 2.1)	78.9±0.2 (↑ 3.0)
	w/ SF (Improv.)	79.2±0.3 (↓ 0.2)	76.1±0.5 (↑ 5.6)	72.8±0.6 (↑ 4.5)	73.4±0.8 (↑ 3.0)	72.6±0.5 (↑ 5.4)	70.2±0.6 (↑ 8.0)	85.9±0.3 (↑ 3.1)	83.4±0.2 (↑ 3.3)	80.5±0.2 (↑ 4.6)
Robust GNNs	RGCN	81.7±0.4	80.1±0.5	78.9±0.5	71.9±0.5	71.5±0.7	69.8±0.6	83.7±0.1	82.9±0.2	81.4±0.2
	w/ PST (Improv.)	82.6±0.6 (↑ 0.9)	81.9±0.5 (↑ 1.8)	79.8±0.6 (↑ 0.9)	73.7±0.6 (↑ 1.8)	73.0±0.4 (↑ 1.5)	72.7±0.5 (↑ 2.9)	85.1±0.2 (↑ 1.4)	85.4±0.3 (↑ 2.5)	84.0±0.3 (↑ 2.6)
	w/ SF (Improv.)	82.7±0.6 (↑ 1.0)	82.0±0.6 (↑ 1.9)	80.6±0.5 (↑ 1.7)	75.0±0.6 (↑ 3.1)	74.8±0.5 (↑ 3.3)	73.1±0.6 (↑ 3.3)	85.6±0.2 (↑ 1.9)	84.9±0.4 (↑ 2.0)	85.1±0.2 (↑ 3.7)
	GNNGuard	82.0±0.6	80.6±0.7	78.5±0.6	71.2±0.4	69.6±0.3	67.4±0.2	86.6±0.1	84.3±0.4	83.6±0.1
	w/ PST (Improv.)	82.3±0.4 (↑ 0.3)	81.1±0.5 (↑ 0.5)	80.3±0.4 (↑ 1.8)	71.7±0.5 (↑ 0.5)	71.1±0.2 (↑ 1.5)	70.5±0.3 (↑ 3.1)	86.5±0.2 (↓0.1)	85.2±0.4 (↑ 0.9)	84.1±0.2 (↑ 0.5)
	w/ SF (Improv.)	83.1±0.7 (↑ 1.1)	**82.6±0.7** (↑ 2.0)	**81.1±0.5** (↑ 2.6)	73.0±0.3 (↑ 1.8)	71.9±0.1 (↑ 2.3)	71.1±0.4 (↑ 3.7)	86.6±0.3 (↑ 0.0)	86.0±0.2 (↑ 1.7)	85.3±0.3 (↑ 1.7)
	GCN-Jaccard	81.4±0.6	80.3±0.4	79.8±0.6	72.1±0.9	72.3±0.7	70.1±0.8	84.0±0.2	82.7±0.1	80.5±0.2
	w/ PST (Improv.)	82.2±0.3 (↑ 0.8)	81.2±0.6 (↑ 0.9)	80.8±0.5 (↑ 1.0)	73.0±0.6 (↑ 0.9)	72.6±0.6 (↑ 0.3)	71.7±0.7 (↑ 1.6)	85.8±0.1 (↑ 1.8)	85.0±0.2 (↑ 2.3)	84.6±0.1 (↑ 4.1)
	w/ SF (Improv.)	82.4±0.4 (↑ 1.0)	81.6±0.8 (↑ 1.3)	81.0±0.6 (↑ 1.2)	**75.3±0.5** (↑ 3.2)	**75.0±0.4** (↑ 2.7)	**74.4±0.5** (↑ 4.3)	86.1±0.1 (↑ 2.1)	85.6±0.0 (↑ 2.9)	84.9±0.1 (↑ 4.4)
	SimP-GCN	80.9±0.5	80.9±0.5	78.9±0.8	73.5±0.7	73.5±0.6	72.6±0.7	86.6±0.2	86.0±0.2	85.3±0.4
	w/ PST (Improv.)	81.6±0.6 (↑ 0.7)	80.9±0.7 (0.0)	80.4±0.6 (↑ 1.5)	73.6±0.6 (↑ 0.1)	73.5±0.8 (0.0)	73.1±0.7 (↑ 0.5)	86.4±0.1 (↓ 0.2)	85.6±0.2 (↓ 0.4)	85.1±0.2 (↓ 0.2)
	w/ SF (Improv.)	82.0±0.8 (↑ 1.1)	81.4±0.6 (↑ 0.5)	80.3±0.8 (↑ 1.4)	75.0±0.4 (↑ 1.5)	74.5±0.8 (↑ 1.0)	73.9±0.7 (↑ 1.3)	86.8±0.1 (↑ 0.2)	86.4±0.1 (↑ 0.4)	**85.8±0.2** (↑ 0.5)
	β-GNN	81.2±0.3	79.7±0.5	76.4±0.5	70.7±0.5	69.4±0.6	63.5±0.7	86.2±0.2	81.5±0.3	78.4±0.2
	w/ PST (Improv.)	82.0±0.3 (↑ 0.8)	80.4±0.6 (↑ 0.7)	78.0±0.5 (↑ 1.6)	72.7±0.6 (↑ 2.0)	72.1±0.6 (↑ 2.7)	71.5±0.4 (↑ 8.0)	86.2±0.1 (↑ 0.0)	85.2±0.2 (↑ 3.7)	85.0±0.3 (↑ 6.6)
	w/ SF (Improv.)	82.5±0.4 (↑ 1.3)	81.0±0.3 (↑ 1.3)	80.2±0.4 (↑ 3.8)	73.9±0.5 (↑ 3.2)	72.1±0.6 (↑ 2.7)	71.8±0.5 (↑ 8.3)	86.8±0.1 (↑ 0.6)	86.0±0.2 (↑ 4.5)	85.6±0.2 (↑ 7.2)
	NoisyGCN	82.0±0.6	81.0±0.3	79.1±0.6	72.0±0.8	71.8±0.4	70.4±0.5	82.8±0.1	81.7±0.1	80.0±0.1
	w/ PST (Improv.)	82.0±0.5 (0.0)	81.2±0.7 (↑ 0.2)	80.1±0.6 (↑ 1.0)	72.0±0.6 (0.0)	71.5±0.5 (↓ 0.3)	70.4±0.9 (0.0)	83.5±0.1 (↑ 0.7)	82.8±0.1 (↑ 1.1)	81.4±0.1 (↑ 1.4)
	w/ SF (Improv.)	82.8±0.5 (↑ 0.8)	82.4±0.6 (↑ 1.4)	81.0±0.5 (↑ 1.9)	72.9±0.6 (↑ 0.9)	72.2±0.7 (↑ 0.4)	71.6±0.9 (↑ 1.2)	84.6±0.2 (↑ 1.8)	83.7±0.1 (↑ 2.0)	82.0±0.2 (↑ 2.0)
	EvenNet	80.4±0.5	78.7±0.7	78.3±1.0	73.8±0.8	73.0±0.5	72.7±0.5	86.2±0.2	85.9±0.2	85.4±0.2
	w/ PST (Improv.)	**83.6±0.5** (↑ 3.2)	81.4±0.7 (↑ 2.7)	80.0±0.6 (↑ 1.7)	74.6±0.7 (↑ 0.8)	73.8±0.8 (↑ 0.8)	73.1±0.7 (↑ 0.4)	86.4±0.1 (↑ 0.2)	85.6±0.2 (↓ 0.3)	85.0±0.1 (↓ 0.4)
	w/ SF (Improv.)	83.4±0.8 (↑ 3.0)	82.1±0.6 (↑ 3.4)	80.8±0.6 (↑ 2.5)	73.7±0.8 (↓ 0.1)	73.0±0.6 (0.0)	72.7±0.7 (0.0)	**87.0±0.1** (↑ 0.8)	**86.7±0.1** (↑ 0.8)	85.5±0.2 (↑ 0.1)

NII, as well as several robust GNNs, such as GCN-Jaccard, EvenNet, and NoisyGCN. These models are evaluated against two typical structure attack methods, Mettack and GraD, across three datasets. The results are shown in Table. III and Table. IV indicate the proposed training strategies can effectively enhance the robustness of both vanilla and robust GNNs in most cases. For instance, GCN w/ PST and GCN w/ SF achieve improvements of 16.9% and 15.2%, respectively, compared to GCN on the Cora dataset under Mettack with a 20% perturbation ratio (highlighted in red in Table. III). Similarly, GAT w/ PST and GCN w/ SF achieve improvements of 10.0% and 13.3%, respectively, compared to GAT on the Citeseer dataset under Mettack with a 20% perturbation ratio. These improvements illustrate that the proposed training strategies can enable vanilla GNNs to achieve robustness comparable to or even exceeding that of robust GNNs such as SimP-GCN and EvenNet.

As for robust GNNs, the proposed training strategies can further enhance their robustness, even though they already possess strong robustness. For instance, NoisyGCN, when applying PST and SF, achieved accuracy improvements of 21.3% and 23.0%, respectively, on the Pubmed dataset under Mettack with a 20% perturbation ratio. Similarly, for EvenNet, PST and SF increase their accuracies by 2.4% and 3.3%, respectively,

TABLE V: Average classification accuracy (± standard deviation) and average training speed (average epochs in seconds) of 10 runs on large-scale graph datasets under PRBCD attacks.

Dataset	ptb(%)	GCN	GADC	EvenNet	Soft Medoid GDC	Soft Median GDC	GCN w/ PST	GCN w/ SF
ogbn-arxiv	clean	66.9+0.32	64.1±0.15	63.2±1.3	57.5±0.24	64.1±0.15	63.8±0.17	**67.0±0.22**
	1%	54.8+0.29	55.6±0.13	36.4±7.92	52.2±0.22	56.9±0.19	58.4±0.20	**58.8±0.16**
	5%	34.6±0.32	45.2±0.17	32.4±4.94	48.0±0.27	47.1±0.21	47.6±0.18	**48.5±0.21**
	10%	29.5±0.58	36.4±0.19	29.2±2.45	**45.4±0.31**	40.8±0.33	43.2±0.24	41.5±0.21
	Training Speed	9.16	8.34	11.1	6.88	7.54	15.1	**18.8**
ogbn-products	clean	73.5±0.08	73.0±0.05	OOM	OOM	64.3±0.0	71.8±0.21	**74.0±0.28**
	1%	63.6±0.10	66.6±0.12			63.0±0.08	64.2±0.17	**69.9±0.30**
	5%	49.5±0.09	58.7±0.17			59.0±0.11	55.4±0.16	**59.6±0.3**
	10%	46.3±0.11	52.5±0.26			**56.9±0.14**	48.6±0.19	54.2±0.26
	Training Speed	2.72	2.82			2.42	3.14	**5.26**

on the Cora dataset under Mettack with a 10% perturbation ratio. Notably, even though EvenNet already exhibits high robustness on Cora, these improvements are achieved.

Although in some cases the performance of the models decreased after applying the training strategies, these declines are minor (0.1% to 0.5%, marked in blue in Table III and Table IV) and are considered normal fluctuations rather than performance degradation caused by the training strategies. As shown in Table. III, EvenNet w/ PST suffers robustness degeneration. Since EvenNet relies on the mechanism that aggregates even-hop neighbors, we attribute this to the adjacency matrix subset of PST that potentially filters even-hop neighbors, which limits EvenNet's mechanism to capture even-hop information, thereby affecting its robustness.

Although model performance occasionally decreased after applying the training strategies, these reductions were minor (0.1% to 0.5%, marked in blue in Table III and Table IV) and are considered normal fluctuations rather than evidence of performance degradation. As shown in Table III, EvenNet with PST exhibits reduced robustness. We attribute this to the PST-selected adjacency matrix subset potentially lacking even-hop neighbors, which interferes with EvenNet's mechanism for capturing even-hop information. Since EvenNet relies on the full adjacency matrix to aggregate even-hop neighbors effectively, the subset selection by PST may limit this capability and thus impact robustness. In summary, we address **RQ1** and **RQ2** by demonstrating that the proposed training strategies are applicable to both vanilla and robust GNNs, and they effectively defend against structural attacks.

C. Efficiency and Scalability Comparison

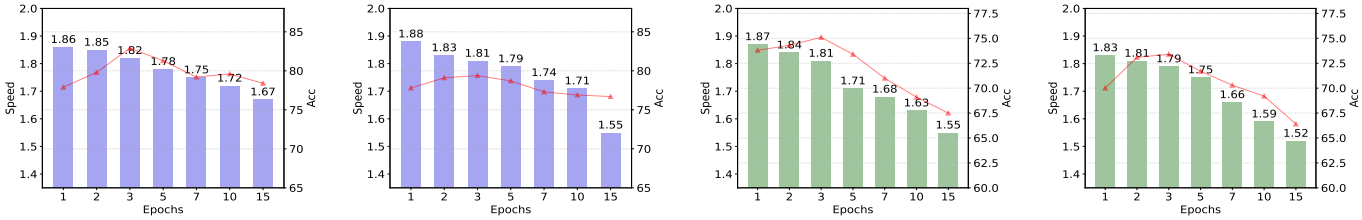
To answer **RQ3**, we evaluate the training speed (epochs/ms) and accuracy (%) of various GNNs before and after applying the proposed strategies on Cora and Citeseer, under Mettack with 20% perturbation rate, as illustrated in Fig. 3. In Fig. 3, we use different markers to distinguish models: circles and squares represent vanilla GNNs and robust GNNs, respectively. Triangles and diamonds denote models with PST and SF, respectively. The model closest to the top-right corner achieves the best accuracy and fastest training speed. From Fig. 3, we observe that purification-based GNNs, i.e., RGCN and GNNGuard, consistently achieve lower accuracy and slower training speed. This is because they rely on purifying the modified structure, which imposes high computational complexity and potentially results in the removal of non-adversarial edges, leading to information loss. In contrast, GNNs with the proposed strategies are always placed above and to the right, achieving higher accuracy and faster training speed compared to robust GNNs (highlighted in squares). This finding indicates that GNNs benefit from improved accuracy and training speed after applying PST or SF. This demonstrates the effectiveness of the proposed training strategies. Additionally, SGC w/ PST and SGC w/ SF consistently achieve the highest accuracy and fastest training speed due to SGC's simple structure, with PST or SF further enhancing both accuracy and speed.

D. The Effectiveness of Disrupting Pairwise Effect

To answer **RQ4** and demonstrate the ability of the proposed training strategies to disrupt pairwise effects, we employ the approximate approach outlined in Appendix. F to measure the approximate value MI_c of the conditional mutual information $I(\hat{\mathbf{A}}; \mathbf{Y}|\mathbf{X})$, during the training of GCN, GCN w/ PST, and GCN w/ SF on modified graphs (Mettack, 20% perturbation ratio). The comparison is shown in Fig. 4. It indicates that when training exceeds 50 epochs, both the performance and MI_c of GCN (red line) drop significantly, indicating that GCN has been poisoned and fails to extract sufficient information from the conditional mutual information for accurate predictions. In contrast, the MI_c of GCN w/ PST and GCN w/ SF steadily increases without being affected, demonstrating their effectiveness in disrupting pairwise effects and thereby achieving robustness.

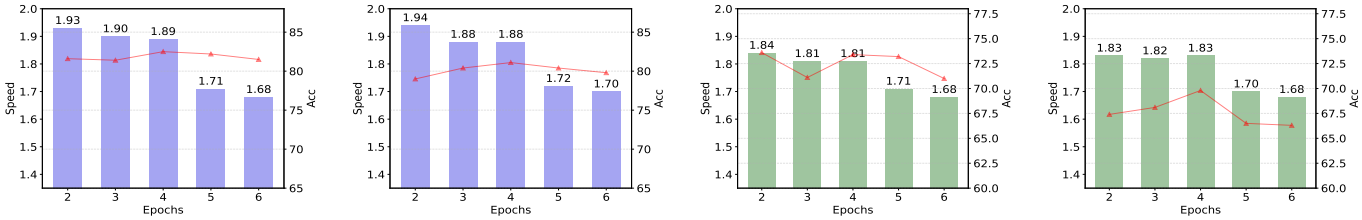
E. Sensitivity Analysis

Hyperparameter Sensitivity. Each training strategies involve only one hyperparameter: fine-tuning epoch and the number of training stages T. We evaluate sensitivity by testing different values for these parameters and analyzing their impact on accuracy and training speed, as shown in Fig. 5 and Fig. 6. The results indicate that the optimal ranges for these parameters are relatively stable. Specifically, the range of fine-tuning epochs is between 3 and 5. Exceeding 5 leads to over-fine-tuning on the modified structure, causing the model to become poisoned, while fewer than 3 epochs



(a) Cora ptb=0 (b) Cora ptb=20% (c) Citeseer ptb=0 (d) Citeseer ptb=20%

Fig. 5: Hyperparameter sensitivity of fine-tuning epochs.



(a) Cora ptb=0 (b) Cora ptb=20% (c) Citeseer ptb=0 (d) Citeseer ptb=20%

Fig. 6: Hyperparameter sensitivity of training stage numbers T.

result in insufficient learning of structural information. For T, the optimal value is consistently 4, as too many training stages increase training time, while too few stages degrade the model's robustness. Overall, compared to the numerous and highly variable hyperparameters of robust GNNs, our strategies require fewer hyperparameters with more stable ranges, offering a clear advantage in hyperparameter tuning.

Intermediate Embedding Sensitivity. We examine the sensitivity of intermediate embeddings to initialization methods, random seeds, the number of training epochs (for PST), and the number of pre-training epochs (for SF).

Fig. 7 shows the training curves of the proposed strategies under commonly used initialization methods, i.e., xavier [55], kaiming [56] and random initialization with three different random seeds. Specifically, taking GCN as an example, Fig. 7 shows training curves of GCN w/ SF and GCN w/ PST on Cora and Citeseer datasets (under Mettack attack with 20% perturbation rate). The horizontal and vertical axes represent the training/pre-training epoch and test accuracy, respectively.

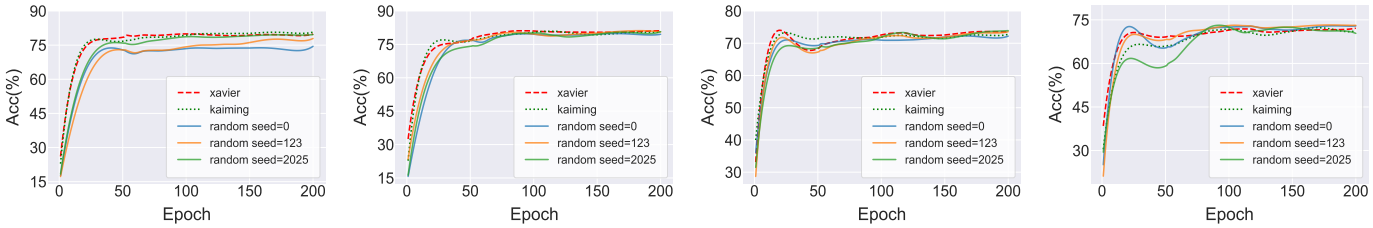
There are two important observations in Fig. 7. First, both GCN w/ PST and GCN w/ SF achieve similar accuracies under different initialization methods and random seeds. Second, all training curves tend to plateau after 100 training or pretraining epochs, indicating that the proposed methods achieve stable performance with sufficient training epochs (i.e., over 100). These observations demonstrate that the intermediate embeddings relied upon by the proposed strategies are not sensitive to initialization methods, random seeds, or the number of training/pre-training epochs.

F. Experiments on Large-scale and Heterophilic Graphs

Evaluation on Large-scale Graphs. Due to memory overflow issues caused by structural attack methods like Mettack on large-scale graphs, we employ PRBCD [57] as the attack method and compare its performance against four robust GNNs capable of scaling to large graphs. The results in Table. V shows that GCN w/ SF and GCN w/ PST consistently achieve either the best or second-best performance across various perturbation ratios on both large-scale datasets. Notably, they also exhibit the fastest training speed, as they do not require purifying the modified structures. Additionally, EvenNet and Soft Medoid GDC encounter out-of-memory (OOM) issues on ogbn-products, since they require at least 70GB of GPU memory, exceeding the 48GB capacity of our experimental device. It is worth noting that the training speed advantage of our strategies is more pronounced on large-scale graphs compared to smaller ones, highlighting the simplicity and effectiveness of our approach.

Evaluation on Heterophilic Graphs. Table. VI compares the robustness of the proposed strategies and baselines on two typical heterophilic graph datasets under Mettack. We report the average classification accuracy (± standard deviation) of 10 runs with different perturbation ratios (ptb). The best and second-best results are highlighted in bold and underlined, respectively. GCN w/ PST achieves the best or second-best performance several times. This indicates that, compared to most robust GNNs, GCN w/ PST has a robustness advantage on heterophilic graphs. However, GCN w/ SF fails to achieve the best or second-best performance. We attribute this to the fact that heterophilic graphs rely more heavily on structural information, while GCN w/ SF depends entirely on node attributes during the pre-training stage and only utilizes structural information during the fine-tuning. This limits its ability to extract relevant information from heterophilic graphs.

In summary, SF demonstrates robustness and efficiency on large-scale graphs, achieving best performance with superior



(a) GCN w/ SF, Cora (b) GCN w/ PST, Cora (c) GCN w/ SF, Citeseer (d) GCN w/ PST, Citeseer

Fig. 7: Training curves of the different initialization methods.

TABLE VI: Robustness comparison on heterophilic graphs.

Dataset	Chameleon			Squirrel		
ptb(%)	0	10	20	0	10	20
GCN	56.3	49.2	40.6	41.2	36.8	34.3
Pro-GNN	56.1	50.2	48.1	**42.0**	37.6	36.1
SimP-GCN	55.6	50.5	46.4	40.9	35.1	32.3
EvenNet	**57.3**	**52.5**	**49.0**	41.3	**38.8**	37.0
STABLE	54.0	49.1	39.9	41.4	35.6	31.5
GADC	56.8	51.0	48.8	41.1	37.7	36.9
Noisy-GCN	56.5	50.1	46.2	40.0	36.1	34.7
GCN w/ PST	57.0	52.0	48.8	40.9	38.1	**37.4**
GCN w/ SF	55.6	51.4	47.2	40.9	37.6	34.7

training speed, while PST achieves satisfactory performance on heterophilic graphs. This answers **RQ5** and verifies the applicability of these strategies in diverse graphs.

G. Difference between SF and PST

It is worth noting that, compared to PST, SF excels in preserving structural integrity, which contributes to its higher accuracy on both clean and modified graphs. However, PST is simpler to implement and achieves faster training speeds on most datasets, with the exception of large-scale graphs. We believe that the multiple training stages of PST are the primary reason for its slower training speed on large-scale graphs. In conclusion, PST is better suited for scenarios where fast training is a priority, whereas SF is more appropriate for applications that demand strong robustness.

VIII. Conclusion

In this work, we uncover the pairwise effect of adversarial structural attacks on graphs through the lens of mutual information. This insight motivates our novel defense strategy: disrupting the pairwise effect to render structural attacks ineffective. To implement this intuition, we propose two simple yet effective training strategies: Progressive Structural Training (PST) and Structural Fine-Tuning (SF), which disrupt the pairwise effect through progressive structure training, and node attributes pre-training followed by structure fine-tuning. These strategies can be easily integrated into a wide range of commonly used GNNs, including robust variants, making them versatile for diverse applications. Extensive experiments on multiple datasets, including large-scale datasets, demonstrate that these strategies not only significantly improve the robustness of various GNN variants against structural attacks but also enhance their training speed significantly.

References

[1] X. Zhang and M. Gan, "Hi-gnn: hierarchical interactive graph neural networks for auxiliary information-enhanced recommendation," *Knowledge and Information Systems*, vol. 66, no. 1, pp. 115–145, 2024.

[2] X. Hu, H. Chen, H. Chen, S. Liu, X. Li, S. Zhang, Y. Wang, and X. Xue, "Cost-sensitive gnn-based imbalanced learning for mobile social network fraud detection," *IEEE Transactions on Computational Social Systems*, 2023.

[3] T. Liu, Y. Wang, R. Ying, and H. Zhao, "Muse-gnn: Learning unified gene representation from multimodal biological graph data," in *Advances in Neural Information Processing Systems*, A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 24 661–24 677. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/4db8a681ae1e58376dc6227978829063-Paper-Conference.pdf

[4] K. Xu, H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, and X. Lin, "Topology attack and defense for graph neural networks: An optimization perspective," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2019, pp. 3961–3967.

[5] Y. Zhu, Y. Lai, K. Zhao, X. Luo, M. Yuan, J. Ren, and K. Zhou, "Binarizedattack: Structural poisoning attacks to graph-based anomaly detection," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 14–26.

[6] P. Kumar and N. Hemachandra, "Edn: A novel edge-dependent noise model for graph data," *arXiv preprint arXiv:2506.11368*, 2025.

[7] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Bylnx209YX

[8] Z. Liu, Y. Luo, L. Wu, Z. Liu, and S. Z. Li, "Towards reasonable budget allocation in untargeted graph structure attacks via gradient debias," in *Advances in Neural Information Processing Systems*, 2022.

[9] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proceedings of the 29th ACM international conference on information & knowledge management*, 2020, pp. 315–324.

[10] H. Zhu, X. Li, P. Zhang, G. Li, J. He, H. Li, and K. Gai, "Learning tree-based deep model for recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 1079–1088.

[11] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 66–74.

[12] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis, "All you need is low (rank) defending against adversarial attacks on graphs," in *Proceedings of the 13th international conference on web search and data mining*, 2020, pp. 169–177.

[13] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, "Node similarity preserving graph convolutional networks," in *Proceedings of the 14th ACM international conference on web search and data mining*, 2021, pp. 148–156.

[14] Y. Zhu, L. Tong, G. Li, X. Luo, and K. Zhou, "Focusedcleaner: Sanitizing poisoned graphs for robust gnn-based node classification," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *International Conference on Learning Representations (ICLR)*, 2017.

[16] K. Li, Y. Liu, X. Ao, J. Chi, J. Feng, H. Yang, and Q. He, "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*, 2022, pp. 925–935.

[17] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Advances in neural information processing systems*, vol. 33, pp. 22 118–22 133, 2020.

[18] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[19] X. Li, Z. Fan, F. Huang, X. Hu, Y. Deng, L. Wang, and X. Zhao, "Graph neural network with curriculum learning for imbalanced node classification," *Neurocomputing*, vol. 574, p. 127229, 2024.

[20] X. Lin, C. Zhou, J. Wu, H. Yang, H. Wang, Y. Cao, and B. Wang, "Exploratory adversarial attacks on graph neural networks for semi-supervised node classification," *Pattern Recognition*, vol. 133, p. 109042, 2023.

[21] Q. Tan, X. Zhang, N. Liu, D. Zha, L. Li, R. Chen, S.-H. Choi, and X. Hu, "Bring your own view: Graph neural networks for link prediction with personalized subgraph selection," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 2023, pp. 625–633.

[22] X. Liu, X. Li, G. Fiumara, and P. De Meo, "Link prediction approach combined graph neural network with capsule network," *Expert Systems with Applications*, vol. 212, p. 118737, 2023.

[23] L. Wei, H. Zhao, Z. He, and Q. Yao, "Neural architecture search for gnn-based graph classification," *ACM Transactions on Information Systems*, vol. 42, no. 1, pp. 1–29, 2023.

[24] Y. Wang, Y. Zhao, N. Shah, and T. Derr, "Imbalanced graph classification via graph-of-graph neural networks," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 2067–2076.

[25] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *stat*, vol. 1050, p. 20, 2017.

[26] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 1024–1034. [Online]. Available: http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.pdf

[27] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2018.

[28] S. Zhang, H. Chen, X. Sun, Y. Li, and G. Xu, "Unsupervised graph poisoning attack via contrastive loss back-propagation," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1322–1330.

[29] J. Zhou, Y. Lai, J. Ren, and K. Zhou, "Black-box attacks against signed graph analysis via balance poisoning," *arXiv preprint arXiv:2309.02396*, 2023.

[30] Y. Zhu, T. Michalak, X. Luo, X. Zhang, and K. Zhou, "Towards secrecy-aware attacks against trust prediction in signed social networks," *IEEE Transactions on Information Forensics and Security*, 2024.

[31] Y. Lai, Y. Zhu, W. Fan, X. Zhang, and K. Zhou, "Towards adversarially robust recommendation from adaptive fraudster detection," *IEEE Transactions on Information Forensics and Security*, 2023.

[32] Y. Zhu, Y. Lai, K. Zhao, X. Luo, M. Yuan, J. Ren, and K. Zhou, "Binarizedattack: Structural poisoning attacks to graph-based anomaly detection," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2022, pp. 14–26.

[33] K. Zhao, H. Zhou, Y. Zhu, X. Zhan, K. Zhou, J. Li, L. Yu, W. Yuan, and X. Luo, "Structural attack against graph based android malware detection," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 3218–3235. [Online]. Available: https://doi.org/10.1145/3460120.3485387

[34] X. Zhang and M. Zitnik, "Gnnguard: Defending graph neural networks against adversarial attacks," *Advances in neural information processing systems*, vol. 33, pp. 9263–9275, 2020.

[35] K. Zhao, Q. Kang, Y. Song, R. She, S. Wang, and W. P. Tay, "Adversarial robustness in graph neural networks: A Hamiltonian energy conservation approach," in *Advances in Neural Information Processing Systems*, New Orleans, USA, Dec. 2023.

[36] S. Zhuang, Z. Wu, Z. Chen, H.-N. Dai, and X. Liu, "Refine then classify: Robust graph neural networks with reliable neighborhood contrastive refinement," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 12, 2025, pp. 13 473–13 482.

[37] S. Ennadir, Y. Abbahaddou, J. F. Lutzeyer, M. Vazirgiannis, and H. Boström, "A simple and yet fairly effective defense for graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, no. 19, 2024, pp. 21 063–21 071.

[38] R. Lei, Z. Wang, Y. Li, B. Ding, and Z. Wei, "Evennet: ignoring odd-hop neighbors improves robustness of graph neural networks," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2024.

[39] N. J. Beaudry and R. Renner, "An intuitive proof of the data processing inequality," *Quantum Information and Computation*, vol. 12, no. 5&6, pp. 432–441, 2012.

[40] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*. PMLR, 2019, pp. 6861–6871.

[41] Z. Chen, Z. Wu, Y. Sadikaj, C. Plant, H.-N. Dai, S. Wang, Y.-M. Cheung, and W. Guo, "Adedgedrop: Adversarial edge dropping for robust graph neural networks," *arXiv preprint arXiv:2403.09171*, 2024.

[42] D. Luo, W. Cheng, W. Yu, B. Zong, J. Ni, H. Chen, and X. Zhang, "Learning to drop: Robust graph neural network via topological denoising," in *Proceedings of the 14th ACM international conference on web search and data mining*, 2021, pp. 779–787.

[43] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[44] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, pp. 127–163, 2000.

[45] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of the third ACM conference on Digital libraries*, 1998, pp. 89–98.

[46] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[47] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *arXiv preprint arXiv:1909.13021*, 2019.

[48] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *International conference on machine learning*. PMLR, 2020, pp. 1725–1735.

[49] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *International Conference on Learning Representations*, 2018.

[50] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, "Robust graph convolutional networks against adversarial attacks," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 1399–1407.

[51] H. I. Aslan, P. Wiesner, P. Xiong, and O. Kao, "β-gnn: A robust ensemble approach against graph structure perturbation," in *Proceedings of the 5th Workshop on Machine Learning and Systems*, 2025, pp. 168–175.

[52] H. Wang, C. Zhou, X. Chen, J. Wu, S. Pan, Z. Li, J. Wang, and P. S. Yu, "Graph structure reshaping against adversarial attacks on graph neural networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 11, pp. 6344–6357, 2024.

[53] S. Liu, J. Chen, T. Fu, L. Lin, M. Zitnik, and D. Wu, "Graph adversarial diffusion convolution," in *International Conference on Machine Learning*, 2024.

[54] Y. Li, W. Jin, H. Xu, and J. Tang, "Deeprobust: A pytorch library for adversarial attacks and defenses," *arXiv preprint arXiv:2005.06149*, 2020.

[55] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[56] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[57] S. Geisler, T. Schmidt, H. Şirin, D. Zügner, A. Bojchevski, and S. Günnemann, "Robustness of graph neural networks at scale," in *Neural Information Processing Systems, NeurIPS*, 2021.