TopFeaRe: Locating Critical State of Adversarial Resilience for Graphs Regarding Topology-Feature Entanglement

Xinxin Fan[1,2] Wenxiong Chen[3,1] Quanliang Jing[1] Chi Lin[3] Shaoye Luo[1,2] Wenbo Song[3,1] Yunfeng Lu[4]
[1]*State Key Laboratory of AI Safety, Institute of Computing Technology, Chinese Academy of Sciences*
[2]*University of Chinese Academy of Sciences, {fanxinxin,jingquanliang, luoshaoye}@ict.ac.cn*
[3]*Dalian University of Technology, c.lin@dlut.edu.cn {cwx1581015, songwenbo}@mail.dlut.edu.cn*
[4]*Beihang University, lyf@buaa.edu.cn*

Abstract

Graph adversarial attacks are usually produced from the two perspectives of topology/structure and node feature, both of them represent the paramount characteristics learned by today's deep learning models. Although some defense countermeasures are proposed at present, they fails to disclose the intrinsic reasons why these two aspects necessitate and how they are adequately fused to co-learn the graph representation. Towards this question, we in this paper propose an adversarial defense approach through locating the graph's critical state of adversarial resilience, resorting to the equilibrium-point theory in the discipline of complex dynamic system (CDS). In brief, our work has three novelties: i) Adversarial-Attack Modeling, i.e. map a graph regime into CDS, and use the oscillation of dynamic system to model the behavior of adversarial perturbation; ii) 2D Topology-Feature-Entangled Function Design for Perturbed Graph, i.e. project graph topology and node feature as two characteristic spaces, and define two-dimensional entangled perturbation functions to represent the dynamic variance under adversarial attacks; and iii) Location of Critical State of Adversarial Resilience, i.e. utilize the equilibrium-point theory to locate the graph's critical state of attack resilience resorting to the perturbation-reflected 2D function. Finally, multi-facet experiments on five commonly-used realistic datasets validate the effectiveness of our proposed approach, and the results show our approach can significantly outperform the state-of-the-art baselines under four representative graph adversarial attacks.

1 Introduction

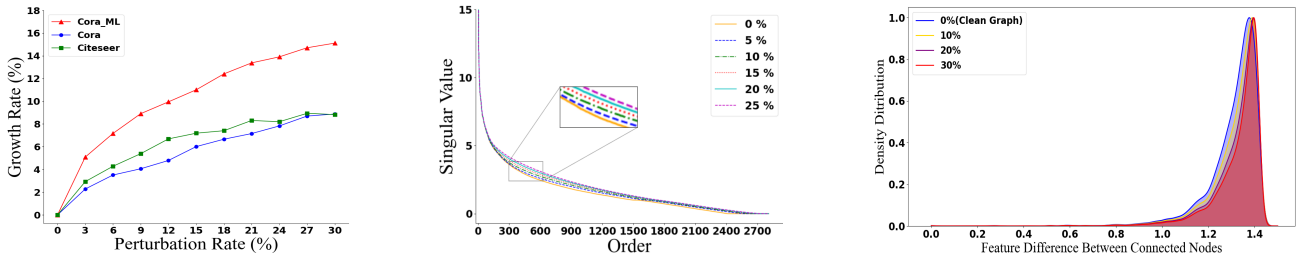
Graph not only demonstrates the linkage/connectivity relationship between homogeneous and heterogeneous nodes, but also bears the attributes (features) among individuals, such as social networks [6], paper-citation network [24], and trade networks [21], etc. Resorting to the graph topology and node features, some typical tasks can be conducted against today's deep-learning models, such as node classification, link prediction, graph alignment, etc. As the technique of deep neural network advances, a set of graph-focused deep learning models (a.k.a. target models) are proposed, including graph neural networks (GNNs) [11, 15, 18], graph convolution network (GCN) [16], graph attention network (GAT) [26], etc.

To date, although these advanced neural networks have greatly enhanced the analytics of graph data, they are vulnerable and become highly susceptible to graph various adversarial attack (GAA) [4, 31, 39] by launching imperceptible perturbations via edge addition/removal, node injection, or/and feature modification, by which the target models are deceived to output incorrect inference. GAA causes severe consequence in security-critical applications, for instance, in loan-networked systems [34], an inauthentic debtor might create multiple transaction records with other authentic users to camouflage himself/herself as a high-credit user [8, 9, 19], as a result, a high reputation/trust score for this inauthentic debtor can be gained to succeed the illegitimate trade under the analytics of conventional GNNs, so do the malicious participants in other interactive (e.g. eCommerce) systems/networks.

In order to mitigate the adverseness, currently two lines of works have been proposed from the viewpoints of adversarial purification and robustness enhancement on GNNs. The former aims to eliminate the perturbations on edges/nodes, or node features to make the graph attack-resilient. The commonly-used strategy is to preprocess the contaminated graphs, for example, the representative works are GCN-SVD [5], GCN-Jaccard [32], and GNNGuard [35]. The latter aims to learn robust representation through designing adequate neural (information-propagation) flow. The popular methods involve GAT [25], RGCN [38], HANG [37], etc. Our work, aiming at exploring the generalized intrinsic critical state of attack resilience from the perspectives of graph topology and node features, belongs to the former category.

As the two basic yet pivotal elements, the topology and feature become the disturbed objects in the procedure of GAA. As well known, the topology apparently demonstrates the

Xinxin Fan and Wenxiong Chen contribute equally.
Corresponding author: Xinxin Fan.



(a) Rank Growth (b) Singular Value (c) Feature Smoothness

Figure 1: Variations of adjacency-matrix rank and singular value, feature smoothness under attack Metattack

correlation among nodes, while the feature indicates the semantics (dis)similarity over two individuals. To make the target models learn inaccurate representation, GAA usually builds linkages between structure-/semantic-dissimilar nodes. In the same vein, our work also leverages such two elements to defeat diversities of adversarial attacks from the standpoint of searching and locating the graph's intrinsic critical state of attack resilience. In brief, our work attempts to explore the dual-resilience domains in terms of graph topology and node feature through resorting to the equilibrium-point (stability) theory of CDS, that is, the process of perturbations under GAA can be recognized as the dynamic variances (oscillations) of CDS over time, and the equilibrium point with the asymptotically-stable status is corresponding to the graph's critical attack-resilient state. Pursuant to this idea, we propose an attack-resilient approach TopFeaRe to resist various GAAs. In a nutshell, three main contributions are involved:

- We exhibit what aspects are influenced by GAA, and theoretically explain why the target models are easily deceived by adversarial perturbations.
- We design topology-feature-entangled function to model the graph's dynamics in terms of topology and feature under adversarial perturbations, and formally analyze what prerequisites are required to satisfy the existence theory of asymptotically-stable equilibrium point.
- We conduct extensive experiments on five real-world datasets under three non-targeted and one targeted popular GAAs to validate the effectiveness of our approach, as well as the rationality of mapping adversarial perturbations into oscillations of CDSs. The experimental results show that our proposed TopFeaRe significantly outperforms the sate-of-the-art baselines.

2 Problem Statement and Our Solution

2.1 Affections of Adversarial Attack

We at first investigate what affections the adversarial attack may bring in. Fig. 1 exhibits the variational tendencies of adjacency-matrix rank, singular value, and feature smoothness under the representative graph adversarial attack Metattack [41], from which we can observe: i) the rank of adjacency matrix goes up apparently as the rate of adversarial perturbation (a.k.a. perturbation rate) enlarges for the three real-world datasets Cora, Cora_ML, and Citeseer; ii) the singular value of adjacency matrix raises as well, especially in the middle segments; and iii) the density distribution of feature difference over connected nodes has a right-transfer phenomenon after adversarial attacks. Pursuant to the experimental results, we know that the GAA can at least causes the three facets of affections in terms of topology and feature. As we know, the currently-existing defense countermeasures are also proposed in consideration of these affections, such as the popular category of preprocessing methods.

2.2 Problem Definition

At present, no matter the preprocessing-based adversarial purification or neural flow-based robustness enhancement, these countermeasures are witnessed to be effective against various GAAs. However, we think the following fundamental questions deserve more inspections and reasonable interpretation.

Question I: Why the existing preprocessing-based methods are effective and whether the original characteristics might be mistakenly deleted. As well known, the preprocessing methods almost purify the adversarial perturbations through removing a certain ratio of contaminated edges/nodes based on the condition of low-rank of adjacency matrix, or high feature-similarity over connected nodes. However, how many edges/nodes ought to be exactly removed is not known, the existing works just depend on pre-defined ratios or similarity thresholds [5, 32], that is to say, there does not exist an explicit referral to guide the ratio and threshold, which might mistakenly under/over-delete the original nodes/edges, leading to the loss of valuable feature. Furthermore, even if the mischievously-added edges or injected nodes are removed, why such operations are effective is unknown.

Question II: Whether to achieve the true meaning of jointly-learning of graph topology and node features. The existing defense methods in general utilize two separate loss

functions to train the topology and feature through setting different hyperparameters, which is not a joint-learning in the true sense. For instance, Pro-GNN [14] defines three loss functions to enhance the robustness, one is to constrain the contribution of the properties of sparsity and low rank from the viewpoint of graph topology, the second is to measure the feature difference over pairwise nodes, and the third is to guide the graph-learning process for specific downstream tasks. Hence, how to realize and guarantee the genuine fusion of jointly-learning regarding topology and feature together, rather than through separate constraints via respective loss functions, is still blank.

Question III: What the relationship between graph data and neural-network learning is. Currently, the two lines of adversarial defenses develop separately, limited studies are dedicated to the relationship between graph data and neural networks. As we know, various graph-focused neural networks, such as GAT [25], RGCN [38], HANG [37], Mid-GCN [13], have manifested the advantage to learn robust embeddings for nodes. However, the more we want to know is, whether the graph data in a proper (equilibrium-point) state can further boost the robustness-learning capacity of such category of neural networks.

Keeping the three fundamental questions in mind, a solution naturally appears, i.e. whether there exists an inherent critical state of attack resilience for each graph, to answer this question, we propose a universal adversarial defense mechanism through exploring such a critical attack-resilient state resorting to the equilibrium-point theory of CDS. In brief, we first map the adversarial perturbations into the oscillation of CDS and measure the adverseness using differential function. Second, we embody topology and feature into two topology-feature-tangled functions to sketch the adversarial perturbations, by which the asymptotically-stable equilibrium point (ASEP) is inferred. Third, referring to such ASEP, we rebuild the contaminated graph to make it attack-resilient.

2.3 Solution Overview

From Fig. 1 we know GAA usually causes the increase of adjacency-matrix rank and singular value, in addition to the right-transfer of feature-density distribution. From the viewpoint of defense, a direct yet valid idea is to descend the rank and singular value through removing the mischievously-added edges and injected nodes, and simultaneously make the feature smoothly through disconnecting the feature-dissimilar nodes that are connected maliciously. However, which dissimilar nodes should be disconnected and how many edges need to be removed remain unclear. Towards this doubt, an idea naturally emerges, namely, for each graph, whether there exists a critical attack-resilient state to resist adversarial attacks, if exists, how to find out it. To this end, regarding the topology and feature together, we employ the equilibrium-point theory of CDS to deduce the asymptotically-stable equilibrium point, by which the necessarily-removed edges are inferred to boost the attack resilience of perturbed graphs.

Remark 2.1. **Relationship Between Oscillations and Perturbations.** CDS can converge into an asymptotically stable equilibrium point (ASEP) after continuous oscillations, such ASEP corresponds to CDS's resilient state that restores to its original state even after long-time oscillations, that is to say, this resilient state implies high robustness. Inspired by this observation, we bridge the connection between oscillations and perturbations from the viewpoint of interpretability.

3 Adversarial Attack Modeling

3.1 Graph-Dynamics Metric

GAA aims to repeatedly modify pivotal edges/nodes, and create connection between feature-dissimilar nodes to fool graph neural networks. Here we refer to such a procedure as adversarial learning, which exactly sketches the dynamic variance of the victim graph under adversarial attacks. Take into consideration the complexity of adversarial perturbations and dependency on concrete topology and feature, it is difficult to give an accurate formula to exactly demonstrate the process of adversarial learning, if not impossible. However, the conventional handling is to model the dynamics of graph, as stated in the literature [10, 17], one node's dynamics can be influenced by its inner self-dynamics and the outer affections by neighboring nodes. Concretely, a node i's dynamics can generally be presented as a linear weighted average of the N state variables each of which is associated with a node i, and the nonlinear influence of other nodes on node i. In reality, the study [10] unveils there indeed exists the nonlinear interactively-affected effect in ecological networks, e.g., the symbiotically connected species networks, such as plants and pollinators or fish and anemone networks. From the standpoint of adversarial perturbations, we also follow this idea of linear and nonlinear perturbations, and provide a universal differential-form definition to sketch the dynamics under continuous adversarial perturbations:

$$\begin{aligned} &\tfrac{d}{dt}\vec{x}(t) = \mathbf{A}\vec{x}(t) + \mathbf{B}\left(-\vec{\phi}\left(\vec{y}(t)\right)\right), \\ &\vec{y}(t) = \mathbf{C}\vec{x}(t) \end{aligned} \tag{1}$$

where $\frac{d}{dt}\vec{x}(t)$ is an N-dimensional vector indicating the state (connectivity) variance of all N nodes during the adversarial perturbations over time t, and $\mathbf{A}\vec{x}(t)$ denotes N-dimensional vector denoting the affected linear perturbations. Given the continuously-perturbed misbehavior on edges in the process of adversarial learning, variable $\vec{y}(t)$ is the output vector standing for the states of all nodes afterward the one-round adversarial perturbations. You can imagine from the angle of

routine forward/back-propagation principle in deep learning, the output vector $\vec{y}(t)$ can in return to influence the dynamics of $\vec{x}(t)$ in a nonlinear manner, i.e. -$\phi(\vec{y}(t))$. Thus, $\frac{d}{dt}\vec{x}(t)$ and $\vec{x}(t)$ are entangled and interplay iteratively, which implies the embodiment of continuous adversarial perturbations on graph topology and node features. The matrices **A**, **B**, **C** are the coefficients to demonstrate the detailed dynamic variations of graph regime in the process of adversarial learning.

Accordingly, the accumulated adverseness can be calculated using the integral from starting time t_0 to $t > 0$, which equivalently corresponds to the aggregation of a period of time continuous perturbations. Therefore, mathematically, for each node i, the metric on time-aware dynamic variation under adversarial perturbations can be formulated as:

$$X_i(t) = \int_{t_0}^{t} x_i(\kappa)\, d\kappa. \tag{2}$$

Of course, the accumulated dynamics for each node are brought by both topology perturbation and feature perturbation, and finally leading to the affection on the whole graph.

3.2 1D Condensed Perturbation Metric

As aforementioned in Eq. (1), the best way is to build a dimension for each node, which can capture the fine-grained perturbations. Nevertheless, such operation may be difficult resulting from two reasons: i) to date there is no principled guidance on how to establish an ASEP-reserved function, referring to the existing knowledge of CDS, in other words, the candidate oscillation-mapping functions need to be figured out manually; and ii) for high-dimensional function, the analysis on whether it has an ASEP would become extremely sophisticated, due to the correlation among different dimensions. Therefore, the common treatment is to use a dimension-reduction function to approximately express the dynamics of the whole graph. Specifically, the edge perturbation can, on one hand, cause the connected nodes' state variation resulting from the change of learnable semantics; on the other hand, each individual itself may have dynamics, e.g. the attribute update itself. Thereby, referring to the individual's own dynamics and pairwise connectivity-caused influence under adversarial perturbations, the universal dynamic-variation of each individual over time can be expressed as:

$$\frac{dx_i}{dt} = F(x_i) + \sum_{j=1}^{N} \mathcal{A}_{ij} G(x_i, x_j), \tag{3}$$

where x_i is the state variable of node i, $F(x_i)$ represents the dynamics of node i itself, and $G(x_i, x_j)$ is the function describing the pairwise connectivity between nodes i and j, A_{ij} denotes the element of the adjacency matrix, indicating the strength of connectivity (a.k.a. semantic-correlation w.r.t. node feature).

Starting from the dimension reduction, we define $\langle \hat{y}_j(x_i) \rangle$ to denote the average dynamics of node i caused by every neighboring node j. Thus, we have that

$$\sum_{j=1}^{N} \mathcal{A}_{ij} G(x_i, x_j) = {s_i}^{in} \langle \hat{y}_j(x_i) \rangle, \tag{4}$$

where ${s_i}^{in} = \sum_{j=1}^{N} \mathcal{A}_{ij}$, and $\langle \hat{y}_j(x_i) \rangle$ represents the weighted average of the connectivity strength $G(x_i, x_j)$ over all in-degree neighbors of node i.

To unify the entangled and sophisticated adversarial perturbations over all pairs of nodes, we employ the classic and commonly-used heterogeneous mean field (HMF) theory [20] to approximate $\langle \hat{y}_j(x_i) \rangle$. Normally, the affection probability $P(j|i)$ of node i placed by its neighboring node j is usually proportional to the j's out-degree ${s_j}^{out}$, i.e. $P(j|i) \propto {s_j}^{out}$. To normalize the affection, the regular way is to regard the normalization factor as the sum of all nodes' out-degrees, i.e. $\sum_{j=1}^{N} {s_j}^{out}$. Thus, the affection probability by node j can defined as $P(j|i) = {s_j}^{out} / \sum_{k=1}^{N} {s_k}^{out}$. Accordingly, the weighted average dynamics of node i can be formulated as,

$$\langle \hat{y}_j(x_i) \rangle_{HMF} = \sum_{j=1}^{N} P(j|i) \hat{y}_j(x_i) = \frac{\frac{1}{N} \sum_{j=1}^{N} {s_j}^{out} \hat{y}_j(x_i)}{\frac{1}{N} \sum_{j=1}^{N} {s_j}^{out}}. \tag{5}$$

As reported in literature [17], if the degree correlation between nodes is small, each individual can be considered to draw from the same distribution, and the average behavior of the graph as a whole can be used to approximate the local behavior of each single node. That is to say, for all nodes we have the approximation equation $\langle \hat{y}_j(x_i) \rangle \approx \langle \hat{y}_j(x_i) \rangle_{HMF}$. As we know, in the interactive networks, such as Twitter and Facebook, the users (nodes) usually have independent activities, leading to small correlation among different individuals, thus, we think the statement above well-matches the reality and the approximation equation is reasonable. The dataset statistics in Table 1 also states this point, i.e., the degree correlations of the five popular datasets are all near to zero. To simplify Eq. (5), we define the mathematical operator Φ as

$$\Phi(\hat{\mathbf{Y}}) = \frac{\mathbf{1}^\top \mathcal{A} \hat{\mathbf{y}}}{\mathbf{1}^\top \mathcal{A} \mathbf{1}} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{A}_{ij} \hat{y}_j}{\sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{A}_{ij}}, \tag{6}$$

where $\hat{\mathbf{Y}} = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_N)^\top$, $\mathbf{1} = (1, 1, \cdots, 1)^\top$. Hence, Eq. (3) can be rewritten as

$$\frac{dx_i}{dt} \cong F(x_i) + {s_i}^{in} \Phi(G(x_i, \mathbf{x})), \tag{7}$$

where $\mathbf{x}=(x_1, \cdots, x_N)^\top$, $G(x_i, \mathbf{x})=(G(x_i, x_1), \cdots, G(x_i, x_N))^\top$, and $\Phi(G(x_i, \mathbf{x}))$ denotes the average edge weight between node i and its neighboring nodes, under HMF theory, it can be further approximated as $G(x_i, \Phi(\mathbf{x}))$. Writing Eq. (7) in vector form, we have

$$\frac{d\mathbf{x}}{dt} = F(\mathbf{x}) + \mathbf{s}^{in} \odot G(\mathbf{x}, \Phi(\mathbf{x})), \tag{8}$$

where $G(\mathbf{x}, \Phi(\mathbf{x})) = (G(x_1, \Phi(\mathbf{x})), \cdots, G(x_N, \Phi(\mathbf{x})))^\top$, $\mathbf{s}^{in} = (s_1^{in}, \cdots, s_N^{in})^\top$, symbol $\odot$ represents the Hadamard product

between vectors, i.e. pairwise elements multiply. Then, substituting the linear mathematical operator Φ into Eq. (8),

$$\begin{aligned}\frac{d\Phi(\mathbf{x})}{dt} &= \Phi(F(\mathbf{x}) + \mathbf{s}^{in} \odot G(\mathbf{x}, \Phi(\mathbf{x}))) \\ &= \Phi(F(\mathbf{x})) + \Phi(\mathbf{s}^{in} \odot G(\mathbf{x}, \Phi(\mathbf{x}))) \\ &\cong F(\Phi(\mathbf{x})) + \Phi(\mathbf{s}^{in}) G(\Phi(\mathbf{x}), \Phi(\mathbf{x}))\end{aligned} . \tag{9}$$

Pursuant to HMF theory [20], we can obtain $\Phi(F(\mathbf{x})) \cong F(\Phi(\mathbf{x}))$ and $\Phi(\mathbf{s}^{in} \odot G(\mathbf{x}, \Phi(\mathbf{x}))) \cong \Phi(\mathbf{s}^{in}) G(\Phi(\mathbf{x}), \Phi(\mathbf{x}))$.

As previously analyzed, an individual's perturbation could render the variations of its neighbors' states, further giving rise to state-change of the whole graph from the global viewpoint. In view of this, the weighted average state of all nodes can be used to sketch the whole graph's state x_{Gra} in a unity way,

$$x_{Gra} = \frac{\langle s^{out} x \rangle}{\langle s^{out} \rangle}. \tag{10}$$

We define a node's state as its degree centrality, stemming from the intuition that the basic yet pivotal degree property plays an important role for the robustness, the subsequent experiments also verify this point. On the other hand, the in-degree of a node can be straightly leveraged to execute perturbation, thus, we take it into account and define a control parameter for the topology-feature-entangled perturbations as

$$\beta_{Gra} = \frac{\langle s^{out} s^{in} \rangle}{\langle s^{out} \rangle}. \tag{11}$$

Substituting $\Phi(\mathbf{x})$=x_{Gra} and $\Phi(\mathbf{s}^{in})$=β_{Gra} into Eq. (9), a one-dimensional state-variation function can be inferred from the viewpoint of whole graph,

$$\frac{dx_{Gra}}{dt} = F(x_{Gra}) + \beta_{Gra} G(x_{Gra}, x_{Gra}). \tag{12}$$

The equation above is a coupled nonlinear ordinary differential equation, $F(x_{Gra})$ and $G(x_{Gra}, x_{Gra})$ respectively reflect the linear and nonlinear properties, here we leverage the 1D form of Michaelis–Menten Equation [1] to represent the nonlinearity, and introduce a constant to present the linearity,

$$\frac{dx_{Gra}}{dt} = \mathcal{B} x_{Gra} + \beta_{Gra} \frac{(x_{Gra})^h}{1 + (x_{Gra})^h}, \tag{13}$$

where $\mathcal{B}$ and h are two adaptive parameters to represent the node-state dynamics under continuous perturbations for different graph regimes. Of course, Michaelis–Menten Equation is just one selection, other functions that can appropriately describe the dynamics of graph perturbations are applied as well, this is because there does not exist guiding principle on how to establish such an dynamics-mapping function. For clarity, we treat the condensed x_{Gra} and β_{Gra} as the dynamic variable and crucial parameter. That is to say, the affection on node-state dynamics by perturbation is subject to β_{Gra}, and for an anticipated critical attack-resilient state $x_0 \in (0, +\infty)$, there exists an instantaneous state β_{Gra} corresponding to $x_{Gra} = x_0$. Easy to understand that, the higher the value of x_{Gra}, the better the graph's attack resilience (robustness). Extremely, if $x_{Gra} \to 0$, it means the entire graph is destroyed completely.

3.3 2D Entangled Perturbation Metric

In essence, both topology and feature play important roles for the graph representation for GNNS. The more sophisticated and powerful adversarial attacks also deceive target models through disturbing the two radical attributes, as shown in Fig. 1, not only do the adjacency-matrix's rank and singular value ascend after attack, but also the distribution of feature difference has a transfer. That is to say, graph topology and node feature are entangled, this is because the edge-connection is intentionally created among feature-dissimilar nodes with the purpose of misleading the node's semantic learning, implying the distinction of feature also influences the edge perturbation during adversarial attack; on the other hand, the topology relying on the edges also disrupts the feature learning. Thus, we ought to redefine the two-factor function into $G(r_i, r_j)$=$H(r_i, r_j, q_j)$, where r_i and q_i can be respectively deemed as graph-topology variable and node-feature variable, $H(r_i, r_j, q_j)$ indicates the entanglement of topology and feature, i.e. the perturbation on node i can potentially affect the variations of edge-connection to node j and the feature-correlation between node j while executing adversarial perturbation. Therefore, we measure the topology-feature-entangled perturbations, and extend the predefined topology-specific dynamics function into two nonlinear ordinary differential functions,

$$\begin{aligned}\frac{dr_i}{dt} &= F(r_i) + \sum_{j=1}^{N} \mathcal{A}_{ij} H(r_i, r_j, q_j) \\ \frac{dq_i}{dt} &= F(q_i) + \sum_{j=1}^{N} \mathcal{A}_{ij} H(q_i, q_j, r_j)\end{aligned}, \tag{14}$$

where r_i and q_i are used to represent the graph-topology dynamics and node-feature dynamics of node i. Pursuant to Eq. (13), we can infer the following formulas:

$$\begin{aligned}\frac{dr_i}{dt} &= -m_i r_{ij} + \frac{\sum_{j=1}^{N} {\gamma_{ij}}^r q_j}{1 + \sum_{j=1}^{N} {\gamma_{ij}}^r q_j} \\ \frac{dq_i}{dt} &= -c_i q_{ij} + \frac{\sum_{j=1}^{N} {\gamma_{ij}}^q r_j}{1 + \sum_{j=1}^{N} {\gamma_{ij}}^q r_j}\end{aligned}, \tag{15}$$

where m_i and c_i represent the degree centrality of node i and the sum of the feature distinctions between node i and its neighboring nodes respectively, ${\gamma_{ij}}^r$ and ${\gamma_{ij}}^q$ are two crucial parameters used to measure the connectivity (correlation) strength between nodes i and j, $\gamma_{ij} = 0$ indicates no connection. Clearly, γ_{ij} is determined by both node degree and

feature discrepancy. Naturally, they can be defined as

$$\begin{aligned} {\gamma_{ij}}^{r} &= \varepsilon_{ij}\frac{1}{\mathcal{K}_i} \\ {\gamma_{ij}}^{q} &= \varepsilon_{ij}\frac{1}{\mathcal{F}_i} \end{aligned}, \tag{16}$$

where ε_{ij}=1 indicates node i is connected to node j, otherwise ε_{ij} = 0. $\mathcal{K}_i$ and $\mathcal{F}_i$ respectively represent the number of neighbors of node i and the sum of feature distinctions between node i and its neighbors.

Referring to the weighted average state (Eq. (10)), we set $q_{Gra} = \frac{\langle \mathcal{F}x \rangle}{\langle \mathcal{F} \rangle}$. The difference between q_{Gra} and r_{Gra} (equal to x_{Gra}) lies in that the dynamic state of each node i is weighted by $\mathcal{F}_i$. Then, considering the entire graph, we have the following two condensed entanglement formulas:

$$\begin{aligned} \sum_{i=1}^{N}\sum_{j=1}^{N}\gamma_{ij}^{r} q_j &\cong \langle \gamma_r \rangle q_{Gra} \\ \sum_{i=1}^{N}\sum_{j=1}^{N}\gamma_{ij}^{q} r_j &\cong \langle \gamma_q \rangle r_{Gra} \end{aligned}. \tag{17}$$

To make the two crucial parameters accurately reflect the graph's perturbations, similarly, we leverage the weighted average manner to process γ_{ij}^{r} and γ_{ij}^{q},

$$\begin{aligned} \langle \gamma_r \rangle &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}{\gamma_{ij}}^{r} \times \mathcal{F}_i}{\sum_{i=1}^{N}\mathcal{F}_i} \\ \langle \gamma_q \rangle &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}{\gamma_{ij}}^{q} \times \mathcal{K}_i}{\sum_{i=1}^{N}\mathcal{K}_i} \end{aligned}, \tag{18}$$

where $\langle \gamma_r \rangle$ and $\langle \gamma_q \rangle$ are obtained by computing the feature distinction-weighted average and the neighbor-count weighted average. In the same way, from the standpoint of dimension reduction, the perturbations in the unified form are reflected by $\langle \gamma_r \rangle$ and $\langle \gamma_q \rangle$. Finally, by calculating the average of the sum of degree centralities of all nodes and the average of the sum of feature distinctions of all nodes to their respective neighboring individuals, the average degree centrality $\mathcal{M}$ and average feature distinction $\mathcal{C}$ are inferred, the node-state dynamic-variation Eq. (15) can be rewritten as

$$\begin{aligned} \frac{dr_{Gra}}{dt} &= -\mathcal{M}r_{Gra} + \frac{\langle \gamma_r \rangle (q_{Gra})^h}{1+\langle \gamma_r \rangle (q_{Gra})^h} \\ \frac{dq_{Gra}}{dt} &= -\mathcal{C}q_{Gra} + \frac{\langle \gamma_q \rangle (r_{Gra})^h}{1+\langle \gamma_q \rangle (r_{Gra})^h} \end{aligned}. \tag{19}$$

Remark 3.1. Our method regards structure topology and node features, and searches for the ASEP for both of them. Eq. (14) presents the topology and feature dynamics under adversarial perturbations. The subsequent Theorem 4.3 proves the asymptotic stability of two-dimensional dynamic-variation mapping function (Eq. (19)) deduced from Eq. (14).

3.4 Generalizability

Although our work provides a generalized theoretical framework for multi-dimensional functions, however, how to find a proper dynamic and ASEP-reserved function is still hard today. Thus, TopFeaRe utilizes HME theory to condense adversarial perturbations of whole graph as a 1D function. For graph data, HMF theory enables to leverage the basic in/out-degree of nodes to approximate the whole graph's information into 1D condensed expression, thus for whatever type of graphs, as long as there exist nodes and edges, HMF can always be applied. Furthermore, although Michaelis-Menten equation is used to design ASEP surface, we can alternatively choose another equation with the only requirement of satisfying the existence of ASEP. Thus, we think our theoretical assumptions can applied into diverse real-world graphs.

4 Theoretical Analytics

4.1 Critical Attack-Resilient State Prerequisite

During adversarial attack, each node would undergoes continuous dynamics, and it is hard to calibrate the node's concrete and instantaneous variation state in the course of continuous perturbations epoch by epoch. Hence, one viable solution is to map the whole graph's dynamic-variation process into a multi-object CDS, in this way, we can analyze the node-state dynamics from a global horizon. Moreover, for CDSs one of most important properties is the stability (a.k.a. equilibrium point). As stated in [7], a dynamic system has three statuses: non-stability, stability, and asymptotic stability, wherein the former implies the system departs from the original status and evolves into a chaotic state; the middle represents the system changes in a bounded scope; the latter indicates the system departs from the original status, and finally converges at an equilibrium point no matter how long it may experience.

We think the three statuses of CDS can well-match the dynamics of graph under the continuous adversarial perturbations. Starting from adversarial defense, the more stable the dynamic system is, the more resilient it would be against adversarial attacks, thus we want to go a step further and anticipate the graph undergoing whatever perturbations can finally evolve into the asymptotically-stable state, corresponding to the ASEP. Toward this end, we introduce the following theorem to warrant the existence of asymptotic stability.

Theorem 4.1. ***(The Existence of Asymptotic Stability State of Graph [7])*** *The dynamic-variance Eq. (1) has an asymptotically-stable equilibrium point if matrix* $\mathbf{A}$ *is Hurwitz, the output* $\vec{y}(t)$ *and its associated nonlinear-mapping input* $\phi\left(\vec{y}(t)\right)$ *satisfy the condition that* $\vec{\phi}\left(\vec{y}(t)\right)^T \cdot \left[\vec{y}(t) - M\vec{\phi}\left(\vec{y}(t)\right)\right] > 0$, *where matrix* $M = diag\left({}^{1}/_{k_1}, \cdots, {}^{1}/_{k_p}\right)$, k_i>0. *Furthermore, given* $\Psi =$

diag $(\chi_1, \cdots, \chi_p)$*, there exists constant* $\chi_i \geq 0$*, such that* $(1+\lambda_k\chi_i) \neq 0$ *for each eigenvalue* λ_k *of matrix* $\mathbf{A}$*, and meanwhile,* $M + (I+s\psi)\,\mathbf{C}\,(sI-\mathbf{A})^{-1}\,\mathbf{B}$ *is strictly positive real. The number of diagonal elements in matrices* M *and* ψ*, as well as the number of eigenvalues of matrix* $\mathbf{A}$*, depends on the dimension of graph.*

Remark 4.1. **Theorem 4.1** demonstrates the generalized guarantee for the graph's asymptotic stability. Towards the condensed topology-feature-entangled dynamics formula, the prerequisite that the perturbed graph can finally converges into an ASEP is figured out, i.e. in what perturbation domain such ASEP can be found out.

Next, we infer the following theorem to present the boundary of perturbation domain in which the ASEP can be located.

Theorem 4.2. ***(Boundary of Perturbation-Domain)*** *To achieve the asymptotic stability equilibrium state of perturbed graph, the boundary of adversarial perturbations must satisfy the inequality* $\frac{(x_{Gra})^{h-1}}{1+(x_{Gra})^{h}} \leqslant k$.

Proof. See Appendix A.

Thereby, by determining the maximum value of $\frac{\phi\left(\vec{y}(t)\right)}{\vec{y}(t)}$, we can identify the perturbation domain to which the element k belongs. If the graph is affected by perturbations from GAAs and k falls into the perturbation domain, then the state of perturbed graph will ultimately converge to the ASEP. To obtain such critical state of adversarial resilience, we set $\frac{dr_{graph}}{dt} = 0$ and $\frac{dq_{graph}}{dt} = 0$, and calculate the topology-feature-entangled equations for r_{Gra} and q_{Gra} w.r.t. the parameters $\langle\gamma_r\rangle$ and $\langle\gamma_q\rangle$,

$$\begin{aligned} r_{Gra}\left(\langle\gamma_r\rangle, \langle\gamma_q\rangle\right) &= \frac{\langle\gamma_r\rangle\langle\gamma_q\rangle + \mathcal{M}\mathcal{C}}{-\mathcal{M}\left(\mathcal{C} - \langle\gamma_r\rangle\right)\langle\gamma_q\rangle} \\ q_{Gra}\left(\langle\gamma_r\rangle, \langle\gamma_q\rangle\right) &= \frac{\langle\gamma_r\rangle\langle\gamma_q\rangle + \mathcal{M}\mathcal{C}}{\mathcal{C}\left(-\mathcal{M} - \langle\gamma_q\rangle\right)\langle\gamma_r\rangle} \end{aligned} . \tag{20}$$

Furthermore, we can adjust the variation of q_{Gra}, such that there exists a variable θ satisfying $r_{Gra} = \theta q_{Gra}$

$$q_{Gra}\left(\langle\gamma_r\rangle, \langle\gamma_q\rangle\right) = \frac{\langle\gamma_r\rangle\langle\gamma_q\rangle + \mathcal{M}\mathcal{C}}{-\theta\mathcal{M}\left(\mathcal{C} - \langle\gamma_r\rangle\right)\langle\gamma_q\rangle} . \tag{21}$$

We recognize the parts $\frac{\langle\gamma_r\rangle q_{Gra}}{1+\langle\gamma_r\rangle q_{Gra}}$ and $\frac{\langle\gamma_q\rangle r_{Gra}}{1+\langle\gamma_q\rangle r_{Gra}}$ of Eq. (19) (h=1) as the nonlinear perturbations. Given that the control parameters $\langle\gamma_r\rangle$ and $\langle\gamma_q\rangle$ can be separated, thus we can obtain the approximate nonlinear parts $\frac{q_{Gra}}{1+q_{Gra}}$ and $\frac{r_{Gra}}{1+r_{Gra}}$. Since the number of elements in matrix M depends on the graph's dimension, for one-dimensional mapping, the matrix M contains only a single element, we decompose the two-dimensional entangled mapping functions into two one-dimensional mapping function with each describing the topology state and feature state respectively. We define $\frac{1}{k_r}$ and $\frac{1}{k_q}$ as the diagonal elements of the one-dimensional mapping function, corresponding to the matrix M of the original N-dimensional graph. According to Theorem 4.2, when k_r and k_q satisfy $k_r \geqslant \frac{1/\theta}{1+(r_{Gra}/\theta)}$ and $k_q \geqslant \frac{\theta}{1+(\theta r_{Gra})}$, respectively, the two-dimensional entanglement function would have an ASEP. Clearly, as $r_{Gra} \to 0$, the minimum values of k_r and k_q can be attained. Therefore, the constants k_r and k_q in Theorem 4.2 belong to the intervals $[1/\theta, \infty)$ and $[\theta, \infty)$ respectively.

In addition, to prove the asymptotic stability of topology-feature-entangled Eq. (19), we offer the following theorem.

Theorem 4.3. ***(Asymptotic Stability of Two-Dimensional Dynamic-Variation Mapping Equation)*** *Decompose the two-dimensional mapping function (Eq. (19)) into two the one-dimensional and analyze the asymptotic stability of each separately. According to the structure of Eq. (1), let* $\mathbf{A}_r$ *and* $\mathbf{B}_r$ *correspond to* $-\mathcal{M}$ *and* $\langle\gamma_q\rangle$ *in the dynamic equations describing the state of graph topology, while* $\mathbf{A}_q$ *and* $\mathbf{B}_q$ *correspond to* $-\mathcal{C}$ *and* $\langle\gamma_r\rangle$ *in the dynamic equations describing the state of node feature. As long as* $\mathbf{A}_r$*,* $\mathbf{B}_r$*,* $\mathbf{A}_q$ *and* $\mathbf{B}_q$ *can form strictly positive real matrix* $\frac{1}{k_r} + (I+s\psi)\,\mathbf{C}\,(sI-\mathbf{A}_r)^{-1}\,\mathbf{B}_r$ *and* $\frac{1}{k_q} + (I+s\psi)\,\mathbf{C}\,(sI-\mathbf{A}_r)^{-1}\,\mathbf{B}_r$*, the two-dimensional dynamic equation exhibits asymptotic stability.*

Proof. See Appendix B

4.2 Surface of Asymptotic Stability

The aforementioned theoretical framework mainly presents that ASEP can be used as an indicator to reconstruct a robust graph, by which our method can purify the contaminated graph to restore its robust state. The two variables r_{Gra} and q_{Gra} reflect the unified perturbation of the graph under adversarial perturbations, and the inferred trajectory points (ASEPs) $\left(r_{Gra}, \langle\gamma_r\rangle, \langle\gamma_q\rangle\right)$ and $\left(q_{Gra}, \langle\gamma_r\rangle, \langle\gamma_q\rangle\right)$ will form a surface in theory to satisfy the asymptotically-stable property.

Note that, the values r_{Gra} and q_{Gra} in Eq. (20) is our theoretical anticipation through bridging the mapping relationship between stability and resilience. To verify the correlation is reasonable and our defined entanglement function can indeed capture the perturbations, we exhibit the actual point distribution using real-world datasets. Towards that goal, the dataset Cora_ML is utilized to output the surface under three popular GAAs, i.e. Metattack [41], CE-PGD [33], and DICE [30]. The dataset statistics information is presented in Table 1. In our experiments, the perturbation rate increases gradually from from 3% to 30% with the increment 3%, and the execution is depicted in Algorithm 1, which aims to sketch an ASEP curved plane that well-matches the different-level adversarial perturbations, that is to say, the proposed theoretical framework provides the prerequisite on how to design a proper ASEP curved plane.

Algorithm 1: ASEP Distribution under Perturbations

Input : Graph $G = (\mathcal{A}, \mathcal{X})$, $\mathcal{A}$ and $\mathcal{X}$ are adjacency and feature matrices

Output : $r_{Gra}, q_{Gra}, \langle \gamma_r \rangle, \langle \gamma_q \rangle$

$r_{Gra} \leftarrow 0, q_{Gra} \leftarrow 0, \langle \gamma_r \rangle \leftarrow 0, \langle \gamma_q \rangle \leftarrow 0$

$S_{out} \leftarrow \text{sum}(\mathcal{A}, \text{axis} = 1)$

for *each* node *in* G **do**

  $x \leftarrow \text{getTopology}(\text{node}, \mathcal{A})$

  $f \leftarrow \text{getFeatureDistinction}(\text{node}, \mathcal{X})$

  $n \leftarrow \text{getNeighbors}(\text{node}, \mathcal{A})$

  $s\_vector.append(x)$

  $f\_vector.append(f)$

  $n\_vector.append(n)$

end

for *each* node *in* G **do**

  $\langle \gamma_r \rangle \leftarrow f\_vector[\text{node}]/n\_vector[\text{node}] + \langle \gamma_r \rangle$

  $\langle \gamma_q \rangle \leftarrow n\_vector[\text{node}]/f\_vector[\text{node}] + \langle \gamma_q \rangle$

end

$r_{Gra} \leftarrow \text{sum}(S_{out} \times s\_vector)/\text{sum}(S_{out})$

$q_{Gra} \leftarrow \text{sum}(f\_vector \times s\_vector)/\text{sum}(f\_vector)$

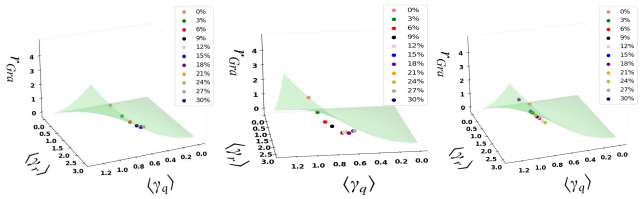
$\langle \gamma_r \rangle \leftarrow \langle \gamma_r \rangle / \text{sum}(f\_vector)$

$\langle \gamma_q \rangle \leftarrow \langle \gamma_q \rangle / \text{sum}(n\_vector)$

return $r_{Gra}, q_{Gra}, \langle \gamma_r \rangle, \langle \gamma_q \rangle$

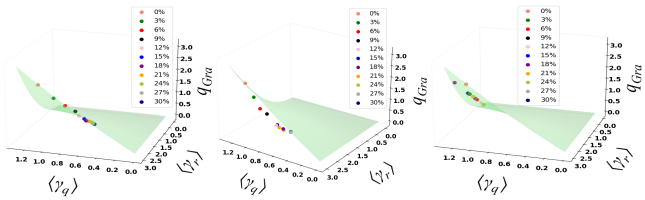
Algorithm 1 takes the adjacency matrix of graph data and its feature matrix as input, and outputs two dynamic variables and key parameters corresponding to the graph data. In the loop, we can obtain the structural state (degree centrality) x of each node, the feature weight f related to its node state, the number of neighboring nodes n of each node, while s_{out} serves as the structural weight of the node state. Finally, the weighted average of $\sum_{j=1}^{N} \gamma_{ij}^{r}$ and $\sum_{j=1}^{N} \gamma_{ij}^{q}$ is calculated. It can be seen that from Eq. 19 the weighted averages of the node states based on structural weights and feature weights are computed separately to obtain dynamic variables r_{Gra} and q_{Gra}, The calculation of the two variables is adapted from Eq. 10. In principle, for a graph regime, no matter degree centrality, betweenness, or closeness, the graph's robustness can be obtained once the ASEP obtained.

The experimental results are drawn in Fig. 2 and Fig. 3, from which we observe that: i) different datasets may have different theoretical surfaces to present the entangled variables r_{Gra} and q_{Gra}; ii) the topology state and feature state of the actual network Cora_ML converge into the inferred 2D theoretical surface of our defined two-dimensional condensed formula, which implies there exists a latent correlation indeed between asymptotic stability and attack resilience, and our bridging is reasonable and effective; and iii) as the rate of adversarial perturbations ascends, the two entangled variables exhibit a decreasing tendency under the three adversarial attacks, which indicates the attack resilience gradually diminishes with the increase of perturbation strength.



(a) Metattack (b) CE-PGD (c) DICE

Figure 2: ASEP surface with r_{Gra} on Cora_ML



(a) Metattack (b) CE-PGD (c) DICE

Figure 3: ASEP surface with q_{Gra} on Cora_ML

5 Implementation

In this section, we detail the implementation on adversarial defense. After the entangled mapping of topology and feature, we know that the dual variables $(\langle \gamma_r \rangle, \langle \gamma_q \rangle)$ strictly affect the robustness. From Fig. 1 we see the GAAs induce the increase of adjacency-matrix's rank and singular values, as well as right-transfer of the density distribution of feature discrepancy. Hence, through this phenomenon, a conclusion can be drawn, that is, the GAAs aim to introduce perturbed edges into the original graph, and these edges are inclined to create links between those nodes that have dissimilar features. To counter that, we need to identify and remove such mischievously-perturbed edges. Towards this purpose, we at first give the following analysis on the basic GCN.

In order to achieve accurate prediction for graph data, GCN [16] usually considers the neighbor's affection to learn node representation. The propagation between layers in neural network is defined as $\hat{H}^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\hat{H}^{(l)}W^{(l)})$, where $\hat{H}$ represents the node features at each layer, $\tilde{A}$ is the sum of adjacency matrix and identity matrix, $\tilde{D}$ denotes the diagonal degree-matrix of $\tilde{A}$, and W is the trainable weight matrix. In the process of aggregating node features, GCN updates the representation vector of each node through l-hop message passing [36]. At $l = 0$, it means the node only aggregates its own feature information, while at $l = 1$, the node collects information from its 1-hop neighbors, and so on. Therefore, GCN relies on nearby nodes to execute representation learning for downstream task. If two nodes are similar, the learning effect of GCN on one node will be better when it transmits information to the other node. Conversely, if they are dissimilar, the information transfer leads to incorrect learning.

For each pair of nodes, we resort to their common neighboring nodes to determine their topology similarity. Instead of only considering the direct neighbors, a more complex approach are implemented, i.e. set $l = 2$. That is to say, we use the two sets of neighboring nodes within 2-hop as parameters to calculate Jaccard similarity, $J(Set_i, Set_j) = \frac{|Set_i \cap Set_j|}{|Set_i \cup Set_j|}$, where Set_i denotes the 2-hop neighboring individuals of node i. On the other hand, the cosine similarity is employed to calculate the pairwise feature similarity via the feature vectors, $C(f_i, f_j) = \frac{f_i \cdot f_j}{\|f_i\| \|f_j\|}$. The larger the Jaccard similarity J and cosine similarity C, the more similar the pairwise nodes i and j. Then, to balance the affections of topology and feature in the procedure of node representation learning, we fuse the neighbor-based similarity and feature-based similarity in a proper way, i.e. assign different weights $\mathcal{W}_J$ and $\mathcal{W}_C$ to them,

$$S(node_i, node_j) = \mathcal{W}_J \times J + \mathcal{W}_C \times C. \tag{22}$$

We calculate the similarity between all pairs of nodes and sort in ascending order in a list. By removing the lower similarity edges at the beginning of the list, we can optimize the graph topology into the attack-resilient state. However, to minimize the risk of mistakenly deleting important edges, the two pivotal variables $\langle \gamma_r \rangle$ and $\langle \gamma_q \rangle$ are referred as the criterion to decide whether to delete or not,

$$\begin{aligned} \textstyle\sum_{j=1}^{N} {\gamma_{ij}}^r|_{(m+1)} > \textstyle\sum_{j=1}^{N} {\gamma_{ij}}^r|_m \\ \textstyle\sum_{j=1}^{N} {\gamma_{ij}}^q|_{(m+1)} > \textstyle\sum_{j=1}^{N} {\gamma_{ij}}^q|_m \end{aligned}. \tag{23}$$

The condition to execute the (m+1)th edge-deletion operation must satisfy the above constraint, otherwise the deleted edge will be restored. The changes in the feature parameters follow a similar logic, satisfying either of the two constraint conditions is sufficient. The implementation procedure is described in Algorithm 2. It takes the adjacency matrix of graph, feature matrix, weights on Jaccard similarity and cosine similarity, an adjustable optimization ratio α as input, and outputs the optimized graph in its desired robust state. First, calculate the weighted similarity between nodes using Eq. 23 and store them in a similarity list in ascending order. Furthermore, within the loop identify and remove undesired edges. Each time an edge with low similarity over nodes is removed, the connection strength between nodes, namely γ_{ij}^r and γ_{ij}^q, will be calculated in each iteration of the loop. If, after removing an edge with low similarity, it is found that the connection strength between these two nodes is actually weaker than it was before the removal. The constraint constructs by Eq. 22. Therefore, the previously-removed but desired edge will be restored, and parameter α determines the optimization ratio.

Algorithm 2: Adversarial Defense Implementation

Input : Graph $G = (\mathcal{A}, \mathcal{X})$, Weights $\mathcal{W}_J$, $\mathcal{W}_C$, rate of adversarial perturbation α
Output : Attack-resilient graph $\hat{G}$

```
Initialize node_i^(γ), node_j^(γ) ← 0, similarity_list ← []
for each node_i, node_j in G do
    Set_i, Set_j ← 1−HopNeighbors(node_i, node_j, A)
    f_i, f_j ← getFeature(node_i, node_j, X)
    J ← JaccardSimilarity(Set_i, Set_j)
    C ← cosineSimilarity(f_i, f_j)
    S ← W_J × J + W_C × C
    similarity_list.append(S)
end
L ← sortAscending(similarity_list)
initial_length ← len(L)
while True do
    Ĝ, node_i, node_j, L ← removeEdge(G, L)
    node_i^(γ), node_j^(γ) ← getValues(A, node_i, node_j)
    if node_i^(γ)|_m < node_i^(γ)|_(m−1) or
      node_j^(γ)|_m < node_j^(γ)|_(m−1) then
        Ĝ, L ← restoreEdge(node_i, node_j)
    end
    if len(L) < α × initial_length then
        break
    end
end
return Ĝ
```

6 Experiment Evaluation

6.1 Configuration

The configurations are detailed in Appendix C: i) **Datasets**: five commonly-used scalable realistic graph datasets: Cora, Cora_ML, Citeseer, Amazon Photo, and PubMed. The statistics are sketched in Table 1; ii) **GAAs:** three non-targeted attacks Metattack [41], CE-PGD [33], and DICE [30], as well as the targeted attack Nettack [40]; iii) **Baselines:** three neural network architectures-focused models: GCN [16], GAT [26], and HANG [37], as well as three purification/preprocessing-based models: GCN-SVD [5], GCN-Jaccard [32], and Mid-GCN [13]; and iv) **Execution Settings:** the running environ-

Table 1: Dataset Statistics

Dataset	Nodes	Edges	Feat.	Classes	Deg. Cor.
Cora_ML	2810	7981	2879	7	-0.0855
Cora	2485	5069	1433	7	-0.0588
Citeseer	2110	3668	3703	6	0.0438
Photo	7487	119043	745	8	0.0247
Pubmed	19717	44325	500	3	0.0395

ment and hyperparameters settings are detailed.

6.2 Performance

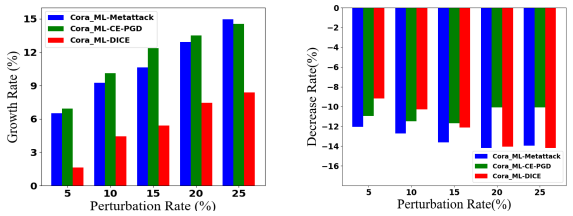
Comparison with Preprocessing-based Baselines. First, we compare our approach to two adjacency-matrix-based preprocessing approaches: GCN-SVD and GCN-Jaccard. Since our TopFeaRe enhances the adversarial resilience through introducing constraints (Eq. (23)) under the guidance of ASEP during the purification of the adjacency matrix, we input the purified graphs into the GCN to execute node-classification tasks. Table 2 and Tables 8-11 (Appendix D) display the accuracy across the five datasets under three non-targeted GAAs.

From the experimental results, we can observe that: i) our TopFeaRe significantly enhances the adversarial resilience compared to GCN-SVD and GCN under all the three adversarial attacks. For example, on Cora_ML under the RAP of 25%, TopFeaRe outperforms the second-best baseline by 16.41%, 1.77%, and 3.26%, respectively. Furthermore, at RAP 0%, which means there are no adversarial perturbations, our method can further promote the adversarial resilience of the clean graph by removing a small number of dissimilar edges. This indicates our proposed method not only performs exceptionally well in adversarial environments, but it also enhances the learning capability of the graph topology under clean conditions. In other words, it is unnecessary to know clean graph and attack signals beforehand, once a graph (even contaminated) is obtained, we can employ existing attacks to draw ASEP surface, then enhance it by such ASEP.

Comparison with Neural Network-Optimized Baselines. We combine our method into GCN, GAT, HANG, and Mid-GCN, and compare to their original version under Metattack attack, that is to say, we in advance modify the adjacency matrix referring to ASEP, then perform the baselines. The experimental results are shown in Table 3 and Tables 12-15 (Appendix D), from which, we can see that our TopFeaRe can significantly boost the adversarial resilience of these neural network-optimized baselines under RAP ranging from 5%-25%, with the accuracy promotion of GCN, GAT, HANG, and Mid-GCN on average [13.095%, 8.67%, 1.06%, 0.43%] on Cora_ML, [18.75%, 7.78%, 4.04%, 1.81%] on Cora, [13.26%, 5.71%, 2.77%, 0.38%] on Citeseer, [4.54%, 9.39%, 1.01%, 1.23%] on Amazon Photo, and [7.30%, 14.86%, 7.29%, 2.26%] on PubMed respectively.

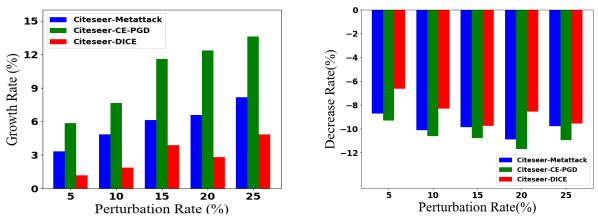
6.3 Ablation Studies

Effect of Topology and Feature. To inspect the effects of topology and feature, we also run a set of ablation experiments from facets: i) only using graph topology-based ASEP to guide the purification of adjacency matrix, called TopRe; and ii) only using node features-based ASEP to guide the purification of adjacency matrix, called FeaRe. The results are listed in Table 4, from which we can observe that for the three



(a) Rank growth under attacks (b) Rank decreased by TopFeaRe

Figure 4: Rank variation of adjacency matrix of Cora_ML.



(a) Rank growth under attacks (b) Rank decreased by TopFeaRe

Figure 5: Rank variation of adjacency matrix of citeseer.

datasets, using the single feature (topology or feature) can obtain similar performance, however, when both are combined, the accuracy can promote by 1.33% to 10.75% for Cora_ML, 1.39% to 16.41% for Cora, and 3.02% to 8.92% for Citeseer under the adversarial attack Metattack. Additionally, we run anther set of experiments with/without two parameters: degree centrality and feature distinction. The performance in Table 5 is obviously better in consideration of the parameters.

HMF vs. Degree Correlations. We run experiments w.r.t. different-level degree correlations (DCs) via pruning on Citeseer, the results in Table 6 show DC has little influence.

6.4 Properties Variation

This subsection studies whether our proposed method has a reverse affection on the adversarial phenomena in Fig. 1.

Rank Variation. Under the three GAAs, we analyzed the rank growth rates and examined the rank descent rates after purifying by TopFeaRe, as shown in Fig. 4 and Fig. 5, the results indicate as RAP gradually increases from 5% to 25%, the rank of adjacency matrix correspondingly raises. This phenomenon suggests adversarial attacks tend to establish connections between nodes, leading to a more complex and dense graph structure. In response, our TopFeaRe significantly reduces the rank through effectively removing the edges over dissimilar nodes. This process not only mitigates the impact of adversarial perturbations, but it also optimizes the graph's topology, making it closer to a robust state.

Singular-Value Variation. To further analyze the changes of singular values, we select Cora_ML and Citeseer datasets

Table 2: Accuracy of node classification on Cora_ML (**Bold**-the best)

GAA	RAP / Mod.	0%	5%	10%	15%	20%	25%
Meta ttack	GCN-SVD	0.8271±0.0080	0.8164±0.0084	0.8054±0.0092	0.7857±0.0095	0.6967±0.1935	0.6279±0.1729
	GCN-Jaccard	0.8423±0.0094	0.7896±0.0087	0.7441±0.0123	0.7028±0.0145	0.6552±0.0294	0.5923±0.0471
	TopFeaRe	**0.8524±0.0096** (↑ **1.01%**)	**0.8293±0.0088** (↑ **1.29%**)	**0.8219±0.0114** (↑ **1.65%**)	**0.8150±0.0107** (↑ **2.93%**)	**0.8086±0.0104** (↑ **11.19%**)	**0.7920±0.0128** (↑ **16.41%**)
CE-PGD	GCN-SVD	0.8271±0.0080	0.8229±0.0066	0.8164±0.0080	0.8118±0.0122	0.8110±0.0089	0.8080±0.0055
	GCN-Jaccard	0.8423±0.0094	0.8334±0.0106	0.8248±0.0112	0.8153±0.0086	0.8131±0.0109	0.8117±0.0082
	TopFeaRe	**0.8524±0.0096** (↑ **1.01%**)	**0.8357±0.0092** (↑ **0.23%**)	**0.8327±0.0112** (↑ **0.79%**)	**0.8307±0.0086** (↑ **1.54%**)	**0.8305±0.0092** (↑ **1.74%**)	**0.8294±0.0090** (↑ **1.77%**)
DICE	GCN-SVD	0.8271±0.0080	0.8148±0.0065	0.8000±0.0064	0.7901±0.0089	0.7730±0.0119	0.7590±0.0087
	GCN-Jaccard	0.8423±0.0094	0.8300±0.0092	0.8177±0.0101	0.8061±0.0088	0.7915±0.0135	0.7752±0.0128
	TopFeaRe	**0.8524±0.0096** (↑ **1.01%**)	**0.8354±0.0089** (↑ **0.54%**)	**0.8291±0.0113** (↑ **1.14%**)	**0.8242±0.0103** (↑ **1.81%**)	**0.8160±0.0122** (↑ **2.45%**)	**0.8078±0.0079** (↑ **3.26%**)

Table 3: Accuracy of node classification on Citeseer under combination with Ours using Metattack

Data set	RAP / Mod.	0%	5%	10%	15%	20%	25%
Cite seer	GCN	0.7211±0.0099	0.6661±0.0219	0.6132±0.0325	0.5538±0.0273	0.4935±0.0261	0.4556±0.0318
	TopFeaRe	**0.7228±0.0105** (↑ **0.17%**)	**0.7159±0.0170** (↑ **4.98%**)	**0.7005±0.0162** (↑ **8.73%**)	**0.6885±0.0170** (↑ **13.47%**)	**0.6754±0.0147** (↑ **18.19%**)	**0.6647±0.0189** (↑ **20.91%**)
	GAT	0.7335±0.0084	0.7140±0.0161	0.6876±0.0256	0.6511±0.0301	0.6170±0.0203	0.5960±0.0283
	GAT-Our	**0.7393±0.0127** (↑ **0.58%**)	**0.7274±0.0094** (↑ **1.34%**)	**0.7210±0.0120** (↑ **3.34%**)	**0.7105±0.0099** (↑ **5.94%**)	**0.6991±0.0137** (↑ **8.21%**)	**0.6934±0.0201** (↑ **9.74%**)
	HANG	**0.7406±0.0116**	0.7268±0.0117	0.7075±0.0272	0.6851±0.0156	0.6722±0.0214	0.6678±0.0305
	HANG-Our	0.7399±0.0104 (↓ **0.07%**)	**0.7348±0.0126** (↑ **0.80%**)	**0.7281±0.0118** (↑ **2.06%**)	**0.7160±0.0106** (↑ **3.09%**)	**0.7095±0.0145** (↑ **3.73%**)	**0.7093±0.0110** (↑ **4.15%**)
	MidGCN	0.7452±0.0032	0.7237±0.0067	0.7105±0.0066	0.6584±0.0106	0.6700±0.0052	0.6306±0.0099
	MidGCN-Our	**0.7466±0.0047** (↑ **0.14%**)	**0.7307±0.0027** (↑ **0.70%**)	**0.7127±0.0080** (↑ **0.22%**)	**0.6626±0.0102** (↑ **0.42%**)	**0.6712±0.0054** (↑ **0.12%**)	**0.6349±0.0134** (↑ **0.43%**)

Table 4: Ablation experiment on node classification accuracy under Metattack

Data set	RAP / Mod.	0%	5%	10%	15%	20%	25%
Cora _ML	w/: Feature	0.8512±0.0033	0.8160±0.0045	0.7576±0.0031	0.7182±0.0059	0.6963±0.0087	0.6307±0.0082
	w/: Topology	0.8515±0.0031	0.8150±0.0043	0.7594±0.0030	0.7241±0.0041	0.7011±0.0071	0.6415±0.0081
	w/: Both	**0.8524±0.0096** (↑ **0.09%**)	**0.8293±0.00886** (↑ **1.33%**)	**0.8219±0.0114** (↑ **6.25%**)	**0.8150±0.0107** (↑ **9.09%**)	**0.8086±0.0104** (↑ **10.75%**)	**0.7920±0.0128** (↑ **15.05%**)
Cora	w/: Feature	0.8230±0.0047	0.7864±0.0152	0.7358±0.0116	0.6309±0.0133	0.6282±0.0189	0.6282±0.0189
	w/: Topology	0.8232±0.0044	0.7893±0.0129	0.7364±0.0093	0.6242±0.0112	0.6322±0.0221	0.5451±0.0231
	w/: Both	**0.8326±0.0083** (↑ **0.94%**)	**0.8032±0.0128** (↑ **1.39%**)	**0.7886±0.0131** (↑ **5.22%**)	**0.7807±0.012** (↑ **14.98%**)	**0.7709±0.0126** (↑ **13.87%**)	**0.7920±0.0128** (↑ **16.38%**)
Cite seer	w/: Feature	0.7107±0.0023	0.6857±0.0062	0.6578±0.0115	0.6174±0.0079	0.6192±0.0071	0.5755±0.0092
	w/: Topology	0.6192±0.0071	0.6857±0.0071	0.6857±0.0071	0.6067±0.0076	0.6083±0.0098	0.5566±0.0160
	w/: Both	**0.7228±0.0105** (↑ **1.21%**)	**0.7159±0.017** (↑ **3.02%**)	**0.7005±0.0162** (↑ **1.48%**)	**0.6885±0.017** (↑ **7.11%**)	**0.6754±0.0147** (↑ **5.62%**)	**0.6647±0.0189** (↑ **8.92%**)

Table 5: Accuracy of node classification w/(w/o) degree centrality and feature distinction under Metattack

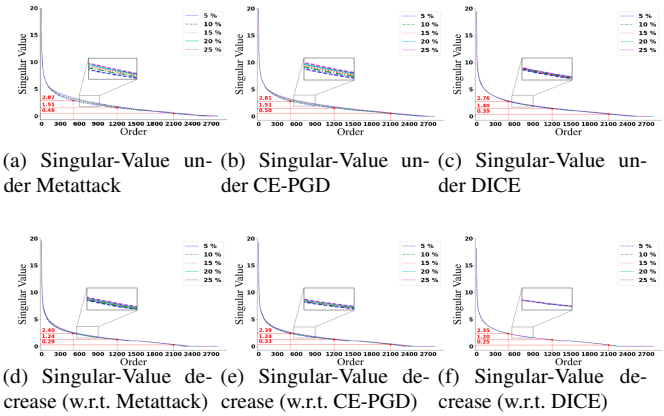
Dataset	RAP	5%	10%	15%	20%	25%
Citeseer	w/o	0.6831±0.0045	0.6560±0.0100	0.6117±0.0078	0.6175±0.0072	0.5870±0.0100
	w/	0.7159±0.0170	0.7005±0.0162	0.6885±0.0170	0.6754±0.0147	0.6647±0.0189

Table 6: Accuracy of node classification w.r.t. different degree correlations on Citeseer

RAP	Accuracy	DCs	RAP	Accuracy	DCs	RAP	Accuracy	DCs
5%	0.7125±0.0036	0.1965	15%	0.7029±0.0060	0.2242	25%	0.7165±0.0086	0.2270
	0.7274±0.0070	0.0147		0.7152±0.0074	0.0281		0.7226±0.0074	0.0312
	0.7267±0.0062	0.0015		0.7104±0.0049	0.0174		0.7137±0.0057	0.0220

to observe the performance under the three attacks. Singular values are part of the spectral properties of a graph, reflecting its structural information. GAAs can alter the spectral properties of a graph, making an originally simple graph more complex to deceive the target models, which usually renders the increase in singular values. As shown in Fig. 6 and Fig. 7, all three attacks introduce unnecessary edges into the perturbed graphs. However, our TopFeaRe successfully reduces the singular values of the adjacency matrices for both datasets, effectively mitigating the impact of adversarial attacks.

Feature-Smoothness Variation. Feature smoothness is a crucial foundation for effective learning from different datasets. Seen from the previous experiments, we know adversarial attacks causing feature non-smoothness can mislead the target models during both training and inference. We present the changes between perturbed graphs and the clean at RAD 25% on Cora_ML and Citeseer datasets, as shown in Fig. 8 and Fig. 9. In clean graphs, feature values are typically concentrated in the regions with low feature distinctions, whereas after adversarial perturbations, this phenomenon is reversed. The purified graphs by TopFeaRe cause the edges in high feature-distinction regions to disperse, while the features become concentrated in the areas with low feature distinctions.



(a) Singular-Value under Metattack (b) Singular-Value under CE-PGD (c) Singular-Value under DICE

(d) Singular-Value decrease (w.r.t. Metattack) (e) Singular-Value decrease (w.r.t. CE-PGD) (f) Singular-Value decrease (w.r.t. DICE)

Figure 6: Singular-Value variation of Cora_ML

6.5 Against Targeted Adversarial Attack

We also use the dataset Citeseer to validate the effectiveness of our proposed method against targeted adversarial attack-Nettack. We compare TopFeaRe with other baseline methods, the experiments are shown in Table 7, from which, we can observe that our proposed TopFeaRe obviously outperforms other baselines across all perturbation counts from 0 to 5. Notably, our TopFeaRe surpasses the popular graph purification methods, such as GCN-SVD and GCN-Jaccard. To sum up, our proposed TopFeaRe demonstrates exceptional performance not only in resisting non-targeted attacks but also in effectively mitigating targeted attacks.

6.6 Computational Complexity

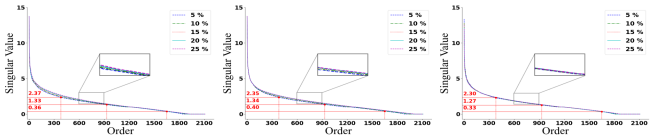
Assume $E(n)$ is the edge function with respect to the node number n, the computational complexity depends on the number of edges, i.e. $O(E(n))$. Furthermore, consider that graph is perturbed pursuant to RAP, thus the total complexity can be formulated as $O(E(n+n*RAP))$. The time overhead is sketched in Fig. 10 as graph size enlarges at RAPs 5%, 15%.

7 Related Work

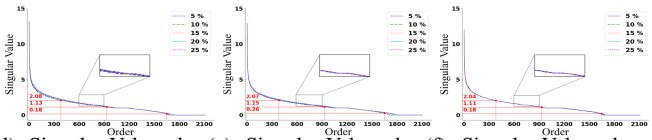
Currently, graph data is facing serious adversarial attacks, primarily due to the complex dependencies (links) between graph regime. For example, from a global perspective, the graph structure can be manipulated to address the collective classification problem [27]. The attack Metattack employs a meta-learning strategy to attack nodes and edges, in addition to adjusting weights, it can rapidly raise the rank of the adjacency matrix and tend to establish connections between different nodes. Xu et al. [33] utilize the popular attack PGD based on negative cross-entropy (CE-PGD) and the strong-attack CW (CW-PGD) to generate topological perturbations. Furthermore, from the perspective of network communities,

Table 7: Accuracy of node classification using Nettack

Data set	Mod. \ Per.	0	1	2	3	4	5
Cite seer	GCN	0.8344±0.0344	0.8309±0.0283	0.8236±0.0210	0.8330±0.0202	0.8375±0.0201	0.8300±0.0162
	GAT	0.7303±0.0095	0.7298±0.0111	0.7322±0.0115	0.7323±0.0150	0.7358±0.0148	0.7277±0.0186
	HANG	0.8390±0.0188	0.8371±0.0223	0.8372±0.0219	0.8356±0.0235	0.8331±0.0229	0.8327±0.0342
	GCN-SVD	0.8297±0.0318	0.8174±0.0233	0.8141±0.0299	0.8186±0.0277	0.8216±0.0217	0.8138±0.0353
	GCN-Jaccard	0.8374±0.0266	0.8356±0.0180	0.8282±0.0189	0.8311±0.0206	0.8269±0.0140	0.8362±0.0240
	TopFeaRe	**0.8617±0.0247** (↑ **2.27%**)	**0.8544±0.0271** (↑ **1.73%**)	**0.8573±0.0247** (↑ **2.01%**)	**0.8588±0.0264** (↑ **2.32%**)	**0.8466±0.0286** (↑ **0.91%**)	**0.8491±0.0278** (↑ **1.29%**)

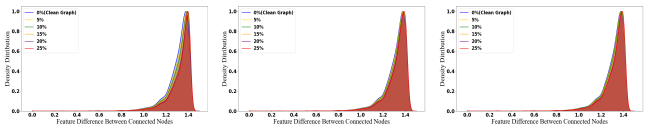


(a) Singular-Value under Metattack (b) Singular-Value under CE-PGD (c) Singular-Value under DICE

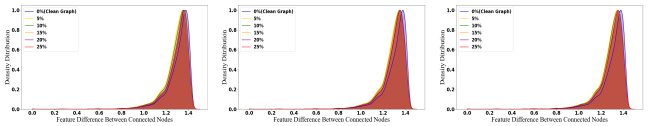


(d) Singular-Value decrease (w.r.t. Metattack) (e) Singular-Value decrease (w.r.t. CE-PGD) (f) Singular-Value decrease (w.r.t. DICE)

Figure 7: Singular-Value variation of Citeseer



(a) Feature-Smoothness under Metattack (b) Feature-Smoothness under CE-PGD (c) Feature-Smoothness under DICE



(d) Feature-Smoothness: Metattack vs. TopFeaRe (e) Feature-Smoothness: CE-PGD vs. TopFeaRe (f) Feature-Smoothness: DICE vs. TopFeaRe

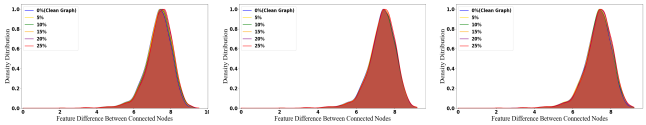
Figure 8: Feature-Smoothness variation of Cora_ML

DICE [30] attacks social networks by deliberately removing internal connections and establishing external connections, thereby reducing the effectiveness of node classification. More seriously, for training graph data, even if it is inaccessible, certain key attributes (such as node degree and subgraph structure) can still be explored through Property Inference Attacks (PIA) [28] and Structural Member Inference Attacks (SMIA) [29]. He et al. [12] proposed a method for stealing graph links from training datasets under black-box access to GNN models. Overall, large graphs are regarded as complex systems with the entanglement of topology and features, and the numerous interactions (information transmission) between nodes make it extremely difficult to defend against persistent adversarial perturbations.

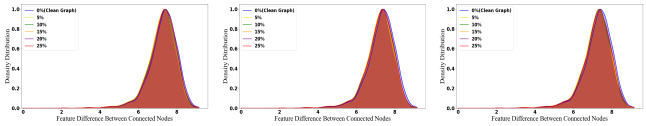
Correspondingly, the adversarial defense mainly focuses on two aspects: graph itself and GNN model. The former enhances adversarial robustness by modifying the graph topology and node features. For example, from a preprocessing perspective, Wu et al. [32] removed dissimilar edges by comparing them with a preset threshold based on Jaccard similarity. GCN-SVD [5], also as a preprocessing method, utilizes Singular Value Decomposition (SVD) to decompose the adjacency matrix of the perturbed graph, obtaining a low-rank approximation that represents a cleaner graph. Another research direction aims to resist adversarial perturbations by enhancing the robustness of GNN (neural-flow) models. For instance, the core idea of ProGNN [14] is to combine the training process of GNNs with the structural properties of the graph, simultaneously learning a new structural graph and a robust GNN on the perturbed graph to ensure that the model can withstand attacks to some extent. At present, the neural ODEs have been successfully applied to GNNs, achieving this by modeling information exchange. References [2] and [23] treat the message passing process as a heat diffusion model, while references [3] and [41] view it as Beltrami diffusion. Additionally, for graph nodes, reference [22] employs an oscillator model to describe nodes and conducts the message passing process under the guidance of coupled ODEs.

8 Conclusion

We gain an in-depth understanding of attack patterns of GAAs and develop corresponding defense mechanism. GAAs tend to disrupt the connection and feature relationships among nodes, leading us to believe that a 1D simplified model is insufficient to adequately describe graph changes in the context.



(a) Feature-Smoothness under Metattack (b) Feature-Smoothness under CE-PGD (c) Feature-Smoothness under DICE



(d) Feature-Smoothness: Metattack vs. TopFeaRe (e) Feature-Smoothness: CE-PGD vs. TopFeaRe (f) Feature-Smoothness: DICE vs. TopFeaRe

Figure 9: Feature-Smoothness variation on Citeseer

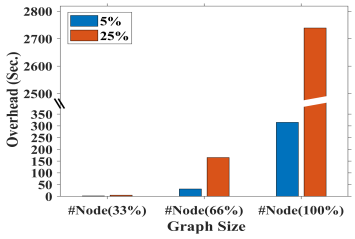


Figure 10: The time overhead as graph (Citeseer) size enlarges

Specifically, for a given graph, we adopt weighting approach to construct a 2D model with two dynamic variables: one to describe topology adversarial resilience and the other to feature adversarial resilience. We demonstrate this 2D simplified model effectively captures the essential characteristics of disrupted graphs under three non-targeted and one targeted attacks, serving as a general paradigm for understanding attack patterns. On the other hand, inspired by the 2D model, we propose a graph purification strategy that effectively defends against various GAAs, showcasing its effectiveness across different datasets and models. In summary, our work sheds light on the potential adversarial defenses from both theoretical and practical angles as the future direction.

Ethical Considerations

In developing the TopFeaRe approach, we have carefully considered the security issues and ethical implications of our research. This research mainly proposes a generalized theory framework and concrete defense algorithm for graph data. Of course, to verify the effectiveness of our proposed method, we also introduce some existing adversarial attacks, which may potential bring security risk to the public community. However, we have adopted several countermeasures to ensure that our research are conducted ethically and responsibly.

Stakeholders and Potential Impact. The key stakeholders involved in our research include data owner, adversarial attacker and defender, and the wider public who might be potentially affected by misuse of these techniques. First, consider from the angle of personal privacy, according to the descriptions and statistics of the five used datasets in this paper, they are open-sourced and not straightly related to personal information. For example, the datasets Cora_ML, Cora are scientific publication-citation networks, the citation links between scientific publications reflect the academic correlation between/within different research subjects/directions, without the disclosure of personal sensitive data. Second, even some graph adversarial attacks are misused, we also introduce some more advanced mitigations to resist these attacks, also our benchmark including the source codes for the mentioned adversarial attacks and defense methods has been released at Github. According the research results, the defenders can enhance the robustness of graph/network-focused systems or platforms. Third, to mitigate these risks, we also comprehensively analyze four representative graph adversarial attacks and six state-of-the-art defense mechanisms, especially analysis on our TopFeaRe from the viewpoint of theory, in addition to emphasizing the need for responsible use of the mentioned adversarial attacks and mitigations, explicitly discourage any unethical applications.

Responsible Disclosure and Concerns. Graph adversarial attacks (GAAs) will bring security risk for graph/network-based systems or platforms, thus, the GAAs may potentially cause severe ethical challenges to many graph-analytics domains, ranging from social media, chemistry (molecule structure), finance (trading network), biology, etc. To address this concerns, we establish a restricted experimental environment to perform GAAs and defense studies, only allowing the research team to access the benchmark in the process of analyzing the detailed vulnerabilities of defense approaches, which minimizes the risks of disclosing the inherent vulnerabilities of existing works and our approach, regarding the potential misuse by attackers. Our institute and universities also provide psychological services for the research team.

Balancing Potential Benefits and Harms. At present, the imperceptible perturbations by GAAs could render the deep learning inference results totally different, which would bring serious consequence to many realistic applications. For example, the deep learning models are currently employed to predict new molecule structure in biopharmaceutics, the incorrect prediction output may yield an unfunctional medicine, or even cause danger to life, thereby, it becomes an extremely important issue on how to design a resilient/robust deep learning model or creature an optimal molecule structure to enhance the model's recognition capability even some disturbed substructures exist. Surrounding this goal, our work proposes a resilience-enhancing approach from the perspective of stability theory, with the purpose of finding out the critical state of adversarial resilience for each graph regime. In this sense, our work can assist to pinpoint the intrinsic stable graph structure,

which is what the many research domains are anticipating. In view of this, our work's research value outweighs the potential harms, even though such critical state of graph regime can be manipulated in return by attacker. Furthermore, from anther viewpoint, for the global digital commerce, through identifying the potential at-risk nodes (inauthentic merchants), our research can validly prevent the economic loss in the process of digital trading.

Open Science

In our work, all the experiments of ours and baselines are performed in an open-source benchmark for adversarial attack and defense, thus, the experimental results are reproducible.

Data Collection and Use. The five datasets used in our experiments, i.e. Cora_ML, Cora, Citeseer Amazon Photo, and PubMed are all open-sourced and commonly-used in academic domain. The five datasets are only used for the experiment evaluation and performance analytics in this paper.

Code and Datasets. We open-source code and datasets at Zenodo: https://doi.org/10.5281/zenodo.17920431.

Acknowledgments

This work has been supported by Strategic Priority Research Program of CAS under Grant No. XDB0680301.

References

[1] Uri Alon. Design principles of biological circuits. *Febs J*, 277:11, 2007.

[2] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International conference on machine learning*, pages 1407–1418. PMLR, 2021.

[3] Benjamin Chamberlain, James Rowbottom, Davide Eynard, Francesco Di Giovanni, Xiaowen Dong, and Michael Bronstein. Beltrami flow and neural diffusion on graphs. *Advances in Neural Information Processing Systems*, 34:1594–1609, 2021.

[4] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR, 2018.

[5] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos 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*, pages 169–177, 2020.

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

[7] Xinxin Fan, Wenxiong Chen, Mengfan Li, Wenqi Wei, and Ling Liu. Adverseness vs. equilibrium: Exploring graph adversarial resilience through dynamic equilibrium. *arXiv preprint arXiv:2505.14463*, 2025.

[8] Xinxin Fan, Ling Liu, Mingchu Li, and Zhiyuan Su. Grouptrust: Dependable trust management. *IEEE Trans. Parallel Distributed Syst.*, 28(4):1076–1090, 2017.

[9] Xinxin Fan, Ling Liu, Rui Zhang, Quanliang Jing, and Jingping Bi. Decentralized trust management: Risk analysis and trust aggregation. *ACM Comput. Surv.*, 53(1):2:1–2:33, 2021.

[10] Jianxi Gao, Baruch Barzel, and Albert-László Barabási. Universal resilience patterns in complex networks. *Nature*, 530(7590):307–312, 2016.

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

[12] Xinlei He, Jinyuan Jia, Michael Backes, Neil Zhenqiang Gong, and Yang Zhang. Stealing links from graph neural networks. In *30th USENIX Security Symposium*, pages 2669–2686, 2021.

[13] Jincheng Huang, Lun Du, Xu Chen, Qiang Fu, Shi Han, and Dongmei Zhang. Robust mid-pass filtering graph convolutional networks. In *Proceedings of the ACM Web Conference 2023*, page 328–338, 2023.

[14] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 66–74, 2020.

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

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

[17] Prosenjit Kundu, Hiroshi Kori, and Naoki Masuda. Accuracy of a one-dimensional reduction of dynamical systems on networks. *Physical Review E*, 105(2):024305, 2022.

[18] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[19] Yunfeng Lu, Xinxin Fan, and Quanliang Jing. Taeffect: Quantifying interaction risks in trust-enabled communication systems. *Int. J. Commun. Syst.*, 36(4), 2023.

[20] Romualdo Pastor-Satorras, Claudio Castellano, Piet Van Mieghem, and Alessandro Vespignani. Epidemic processes in complex networks. *Reviews of modern physics*, 87(3):925–979, 2015.

[21] Hao Qian, Hongting Zhou, Qian Zhao, Hao Chen, Hongxiang Yao, Jingwei Wang, Ziqi Liu, Fei Yu, Zhiqiang Zhang, and Jun Zhou. Mdgnn: Multi-relational dynamic graph neural network for comprehensive and dynamic stock investment prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14642–14650, 2024.

[22] T Konstantin Rusch, Ben Chamberlain, James Rowbottom, Siddhartha Mishra, and Michael Bronstein. Graph-coupled oscillator networks. In *International Conference on Machine Learning*, pages 18888–18909. PMLR, 2022.

[23] Matthew Thorpe, Tan Nguyen, Hedi Xia, Thomas Strohmer, Andrea Bertozzi, Stanley Osher, and Bao Wang. Grand++: Graph neural diffusion with a source term. *ICLR*, 2022.

[24] Vincent A Traag and Jeroen Bruggeman. Community detection in networks with positive and negative links. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 80(3):036115, 2009.

[25] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[26] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[27] Binghui Wang and Neil Zhenqiang Gong. Attacking graph-based classification via manipulating the graph structure. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2023 – 2040, 2019.

[28] Xiuling Wang and Wendy Hui Wang. Group property inference attacks against graph neural networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 2871–2884, 2022.

[29] Xiuling Wang and Wendy Hui Wang. Subgraph structure membership inference attacks against graph neural networks. *Proceedings on Privacy Enhancing Technologies*, 2024.

[30] Marcin Waniek, Tomasz P Michalak, Michael J Wooldridge, and Talal Rahwan. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2):139–147, 2018.

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

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

[33] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. *arXiv preprint arXiv:1906.04214*, 2019.

[34] Shuo Yang, Zhiqiang Zhang, Jun Zhou, Yang Wang, Wang Sun, Xingyu Zhong, Yanming Fang, Quan Yu, and Yuan Qi. Financial risk analysis for smes with graph-based supply chain mining. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 4661–4667. ijcai.org, 2020.

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

[36] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):249–270, 2020.

[37] Kai Zhao, Qiyu Kang, Yang Song, Rui She, Sijie Wang, and Wee Peng Tay. Adversarial robustness in graph neural networks: A hamiltonian approach. *Advances in Neural Information Processing Systems*, 36, 2024.

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

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

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

[41] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

A Boundary of Perturbation-Domain

Proof. According to Theorem 4.1, there exists a diagonal matrix M such that the perturbed output $\vec{y}(t)$ and its associated non-linear mapped input $\phi\left(\vec{y}(t)\right)$ satisfy the condition $\vec{\phi}\left(\vec{y}(t)\right)^T \cdot \left[\vec{y}(t) - M\vec{\phi}\left(\vec{y}(t)\right)\right] \geqslant 0$. The diagonal elements of the diagonal M are influenced by $\phi\left(\vec{y}(t)\right)$. Using Eq. (13), $\frac{(x_{Gra})^h}{1+(x_{Gra})^h}$ is deemed as the nonlinear perturbation input, and correspondingly $\phi\left(\vec{y}(t)\right) = \frac{(x_{Gra})^h}{1+(x_{Gra})^h}$ and $\vec{y}(t) = x_{Gra}$. Given that $\phi\left(\vec{y}(t)\right)^T > 0$ and $\vec{y}(t) - M\vec{\phi}\left(\vec{y}(t)\right) \geqslant 0$, the inequality $\frac{\phi\left(\vec{y}(t)\right)}{\vec{y}(t)} = \frac{(x_{Gra})^{h-1}}{1+(x_{Gra})^h} \leqslant k$ is inferred.

B Asymptotic Stability of Two-Dimensional Dynamic-Variation Mapping Equation

Proof. According to the definition in Eq. (19), it is clear that $-\mathcal{M}$ with a single element, its eigenvalue is simply the element itself. Therefore, the eigenvalue of $\mathbf{A}_r$ is $-\mathcal{M}$, which implies that $\mathbf{A}_r$ is a Hurwitz matrix. At this point, there exists $\chi_r \geq 0$, and when the condition $(1 - \mathcal{M}\chi_r) \neq 0$ is satisfied, the one-dimensional dynamic equation can be computed as

$$\begin{aligned} \widetilde{G}_r(s) &= -\mathcal{M} + (I + s\chi_r)(sI - \mathbf{A}_r)^{-1}\mathbf{B}_r \\ &= \frac{1}{k_r} + (1 - \chi_r\mathcal{M})(s + \mathcal{M})^{-1}\langle\gamma_r\rangle + \chi_r\langle\gamma_r\rangle, \end{aligned} \tag{24}$$

where $s = \sigma + \omega j$ and the complex unit $j = \sqrt{-1}$. First, it is clear that $\widetilde{G}_r(s) + \widetilde{G}_r(-s)$ is not identically zero. Moreover, since $\mathcal{M}$ is positive, it can be computed that

$$\widetilde{G}_r(\omega j) + \widetilde{G}_r^T(-\omega j) = \frac{2}{k_r} + 2\langle\gamma_r\rangle\frac{\mathcal{M} + \chi_r\omega^2}{\mathcal{M}^2 + \omega^2}. \tag{25}$$

Similarly, it can be obtained that

$$\widetilde{G}_q(\omega j) + \widetilde{G}_q^T(-\omega j) = \frac{2}{k_q} + 2\langle\gamma_q\rangle\frac{\mathcal{C} + \chi_q\omega^2}{\mathcal{C}^2 + \omega^2}. \tag{26}$$

Therefore, for all real numbers ω, the matrices $\widetilde{G}_r(s)$ and $\widetilde{G}_q(s)$ are positive definite and strictly positive real.

C Configuration

Datasets. The experiments are performed on five commonly-used scalable realistic graph datasets: Cora, Cora_ML, Citeseer, Amazon Photo, and PubMed.

Baselines. To evaluate our approach, we employ four types of graph adversarial attacks, including three non-targeted attacks Metattack [41], CE-PGD [33], and DICE [30], as well as the targeted attack Nettack [40]. For the target models, we incorporate three neural network architectures-focused and two preprocessing-based models: i) **GCN [16]**: As the representative of GNNs, it has been designed to extract features in various downstream tasks; ii) **GAT [26]**: By utilizing multiple attention layers to dynamically adjust the weights of neighboring nodes, it can better learn the relationships and feature information; iii) **HANG [37]**: It is a recently-proposed graph neural flow method, and resorting to Hamiltonian neural flows to learn robust node embedding; iv) **GCN-SVD [5]**: It serves as a preprocessing method to defend against adversarial attacks by improving the robustness of GCN through low-rank approximation of the perturbed graph; v) **GCN-Jaccard [32]**: By removing edges with low Jaccard similarity smaller than a predefined threshold, the graph topology is optimized to minimize the distinctions in node features caused by adversarial attacks; and vi) **Mid-GCN [13]**: It designs a mid-pass filtering GCN to leverage mid-frequency signals (i.e. Laplacian eigenvalue around 1) to mitigate the adversarial attacks.

Execution Settings. For non-targeted attacks, we set the rate of adversarial perturbation (RAP), i.e. the ratio of perturbed edges over all edges, to range from 0% to 25% in steps of 5%. For targeted attacks, we select those nodes whose degree is greater than 10 in the test set for perturbation, with the number of perturbations ranging from 1 to 5 in steps of 1. For each graph, we randomly select 10% of the nodes for model training, 10% for validation, and 80% for testing. The accuracy of node classification will be averaged over ten-time experiments. For baselines' settings, GCN [16], GAT [26], and HANG [37] all use their default parameters. For GCN-SVD [5], we choose the optimal rank reduction number from {20, 40, 60, 80, 100}. For GCN-Jaccard [32], the threshold is selected from {0.01, 0.02, 0.04, 0.06, 0.08, 0.1} to find the optimal value. For our TopFeaRe, α is chosen from {0.75, 0.8, 0.85, 0.9, 0.95, 0.99} for the optimal value, with $\mathcal{W}_J$ and $\mathcal{W}_C$ setting to 0.3 and 0.7. The hardware running environment is: NVIDIA A800 80GB PCIe.

D Comparison with Baselines

Tables 8-11 provide the comparative experiments between ours and those baselines using preprocessing-based purification on datasets Cora, Citeseer, Amazon Photo, and PubMed. Tables 12-15 provide the comparative experiments between ours and those baselines using network-optimized defense on datasets Cora_ML, Cora, Amazon Photo, and PubMed.

Table 8: Accuracy of node classification on Cora

GAA	RAP / Mod.	0%	5%	10%	15%	20%	25%
Metattack	GCN-SVD	0.7801±0.0091	0.7622±0.0145	0.7478±0.0103	0.7090±0.0250	0.6849±0.0216	0.5800±0.1602
	GCN-Jaccard	0.8221±0.0099	0.7819±0.0133	0.7561±0.0134	0.7066±0.0288	0.6878±0.0179	0.6491±0.0248
	TopFeaRe	**0.8326±0.0083** (↑ **1.05%**)	**0.8032±0.0128** (↑ **2.13%**)	**0.7886±0.0131** (↑ **3.25%**)	**0.7807±0.0120** (↑ **7.17%**)	**0.7709±0.0126** (↑ **8.31%**)	**0.7521±0.0159** (↑ **10.30%**)
CE-PGD	GCN-SVD	0.7801±0.0091	0.7682±0.0067	0.7624±0.0130	0.7555±0.0120	0.7488±0.0072	0.7440±0.0092
	GCN-Jaccard	0.8221±0.0099	0.8188±0.0102	0.8179±0.0092	0.8134±0.0090	0.8093±0.0067	0.8091±0.0096
	TopFeaRe	**0.8326±0.0083** (↑ **1.05%**)	**0.8194±0.0105** (↑ **0.06%**)	**0.8186±0.0081** (↑ **0.07%**)	**0.8166±0.0061** (↑ **0.32%**)	**0.8158±0.0099** (↑ **0.65%**)	**0.8154±0.0061** (↑ **0.63%**)
DICE	GCN-SVD	0.7801±0.0091	0.7235±0.0080	0.7096±0.0121	0.6898±0.0165	0.6775±0.0063	0.6609±0.0094
	GCN-Jaccard	0.8221±0.0099	0.8102±0.0102	0.8024±0.0100	0.7894±0.0094	0.7771±0.0115	0.7678±0.0078
	TopFeaRe	**0.8326±0.0083** (↑ **1.05%**)	**0.8152±0.0080** (↑ **0.50%**)	**0.8084±0.0080** (↑ **0.60%**)	**0.7970±0.0074** (↑ **0.76%**)	**0.7881±0.0104** (↑ **1.10%**)	**0.7807±0.0091** (↑ **1.29%**)

Table 9: Accuracy of node classification on Citeseer

GAA	RAP / Mod.	0%	5%	10%	15%	20%	25%
Metattack	GCN-SVD	0.6831±0.0126	0.6686±0.0149	0.6456±0.0216	0.5634±0.1718	0.5832±0.0254	0.5708±0.0285
	GCN-Jaccard	**0.7264±0.0126**	0.7098±0.0148	0.6802±0.0197	0.6469±0.0264	0.6227±0.0161	0.5956±0.0310
	TopFeaRe	**0.7264±0.0135** (↑ **0%**)	**0.7159±0.0170** (↑ **0.61%**)	**0.7005±0.0162** (↑ **2.03%**)	**0.6885±0.0170** (↑ **4.16%**)	**0.6754±0.0147** (↑ **5.27%**)	**0.6647±0.0189** (↑ **6.91%**)
CE-PGD	GCN-SVD	0.6831±0.0126	0.6773±0.0106	0.6763±0.0127	0.6960±0.0067	0.6875±0.0142	0.6842±0.0119
	GCN-Jaccard	**0.7264±0.0126**	0.7184±0.0140	0.7167±0.0126	0.7137±0.0128	0.7136±0.0134	0.7111±0.0144
	TopFeaRe	**0.7264±0.0135** (↑ **0%**)	**0.7226±0.0137** (↑ **0.42%**)	**0.7226±0.0114** (↑ **0.59%**)	**0.7221±0.0113** (↑ **0.84%**)	**0.7185±0.0111** (↑ **0.49%**)	**0.7146±0.0111** (↑ **0.35%**)
DICE	GCN-SVD	0.6831±0.0126	0.6555±0.0161	0.6461±0.0230	0.6232±0.0155	0.6165±0.0178	0.5978±0.0213
	GCN-Jaccard	**0.7264±0.0126**	**0.7185±0.0136**	0.7081±0.0123	0.6964±0.0141	0.6834±0.0137	0.6777±0.0123
	TopFeaRe	**0.7264±0.0135** (↑ **0%**)	0.7126±0.0156 (↓ **0.59%**)	**0.7116±0.0141** (↑ **0.35%**)	**0.7008±0.0134** (↑ **0.44%**)	**0.6964±0.0179** (↑ **1.30%**)	**0.6847±0.0128** (↑ **0.70%**)

Table 10: Accuracy of node classification on Amazon Photo

GAA	RAP / Mod.	0%	5%	10%	15%	20%	25%
Metattack	GCN-SVD	0.8713±0.0047	0.8238±0.0235	0.8092±0.0249	0.7651±0.0827	0.6632±0.1165	0.6567±0.1482
	GCN-Jaccard	0.9232±0.0144	0.8327±0.0399	0.8217±0.0169	0.7332±0.1163	0.6344±0.1348	0.6496±0.1505
	TopFeaRe	**0.9243±0.0118** (↑ **0.11%**)	**0.8486±0.0682** (↑ **1.59%**)	**0.8532±0.0338** (↑ **3.15%**)	**0.7754±0.1088** (↑ **1.03%**)	**0.7108±0.1246** (↑ **4.76%**)	**0.6847±0.1278** (↑ **2.80%**)
CE-PGD	GCN-SVD	0.8713±0.0047	0.8756±0.0029	0.8743±0.0094	0.8771±0.0030	0.8672±0.0037	0.8680±0.0042
	GCN-Jaccard	0.9232±0.0144	0.9111±0.0039	0.8998±0.0231	0.8989±0.0043	0.8831±0.0057	0.8406±0.1294
	TopFeaRe	**0.9243±0.0118** (↑ **0.11%**)	**0.9198±0.0143** (↑ **0.87%**)	**0.9091±0.0072** (↑ **0.93%**)	**0.9061±0.0051** (↑ **0.72%**)	**0.8976±0.0076** (↑ **1.45%**)	**0.8957±0.0075** (↑ **2.77%**)
DICE	GCN-SVD	0.8713±0.0047	0.8467±0.0064	0.8358±0.0048	0.8077±0.0163	0.7650±0.0273	0.7377±0.0671
	GCN-Jaccard	0.9232±0.0144	0.9099±0.0173	0.8995±0.0095	0.8917±0.0093	0.8763±0.0156	0.8481±0.0273
	TopFeaRe	**0.9243±0.0118** (↑ **0.11%**)	**0.9189±0.0093** (↑ **0.90%**)	**0.9094±0.0138** (↑ **0.99%**)	**0.8975±0.0113** (↑ **0.58%**)	**0.8943±0.0111** (↑ **1.80%**)	**0.8803±0.0164** (↑ **3.22%**)

Table 11: Accuracy of node classification on PubMed

GAA	RAP / Mod.	0%	5%	10%	15%	20%	25%
Metattack	GCN-SVD	0.8455±0.0007	0.8457±0.0014	0.8457±0.0014	0.8427±0.0009	0.8164±0.0024	0.7887±0.0025
	GCN-Jaccard	0.8685±0.0007	0.7981±0.0057	0.6381±0.0088	0.5588±0.0079	0.5320±0.0062	0.4450±0.0246
	TopFeaRe	**0.8824±0.0054** (↑ **1.39%**)	**0.8672±0.0054** (↑ **2.15%**)	**0.8579±0.0095** (↑ **1.22%**)	**0.8514±0.0110** (↑ **0.87%**)	**0.8365±0.0083** (↑ **2.01%**)	**0.8177±0.0038** (↑ **2.90%**)
CE-PGD	GCN-SVD	0.8455±0.0007	0.8476±0.0016	0.8469±0.0009	0.8448±0.0005	0.8387±0.0005	0.8374±0.0007
	GCN-Jaccard	0.8685±0.0007	0.8539±0.0004	0.8416±0.0004	0.8305±0.0011	0.8261±0.0004	0.8154±0.0016
	TopFeaRe	**0.8824±0.0054** (↑ **1.39%**)	**0.8644±0.0095** (↑ **1.05%**)	**0.8586±0.0038** (↑ **1.17%**)	**0.8549±0.0025** (↑ **1.01%**)	**0.8476±0.0068** (↑ **0.89%**)	**0.8457±0.0050** (↑ **0.83%**)
DICE	GCN-SVD	0.8455±0.0007	0.8408±0.0010	0.8332±0.0010	0.8332±0.0011	0.8258±0.0012	0.8206±0.0016
	GCN-Jaccard	0.8685±0.0007	0.8530±0.0008	0.8351±0.0012	0.8236±0.0012	0.8103±0.0010	0.7926±0.0012
	TopFeaRe	**0.8824±0.0054** (↑ **1.39%**)	**0.8623±0.0060** (↑ **0.93%**)	**0.8579±0.035** (↑ **2.28%**)	**0.8517±0.045** (↑ **1.85%**)	**0.8459±0.0019** (↑ **2.01%**)	**0.8386±0.015** (↑ **1.80%**)

Table 12: Accuracy of node classification on Cora_ML under combination with Ours using Metattack

Data set	RAP / Mod.	0%	5%	10%	15%	20%	25%
Cora_ML	GCN	0.8505±0.0111	0.7915±0.0096	0.7353±0.0152	0.6856±0.0193	0.6333±0.0339	0.5665±0.0471
	TopFeaRe	**0.8516±0.0104** (↑ **0.11%**)	**0.8293±0.0088** (↑ **3.78%**)	**0.8219±0.0114** (↑ **8.66%**)	**0.8150±0.0107** (↑ **12.94%**)	**0.8086±0.0104** (↑ **17.53%**)	**0.7920±0.0128** (↑ **22.55%**)
	GAT	0.8466±0.0108	0.8005±0.0165	0.7590±0.0150	0.7238±0.0172	0.6778±0.0351	0.6174±0.0500
	GAT-Our	**0.8476±0.0097** (↑ **0.10%**)	**0.8213±0.0142** (↑ **2.08%**)	**0.8146±0.0124** (↑ **5.56%**)	**0.8069±0.0085** (↑ **8.31%**)	**0.7960±0.0094** (↑ **11.82%**)	**0.7730±0.0140** (↑ **15.56%**)
	HANG	0.8419±0.0096	0.8287±0.0100	0.8175±0.0105	0.8029±0.0062	0.7927±0.0170	0.7737±0.0218
	HANG-Our	**0.8466±0.0099** (↑ **0.47%**)	**0.8315±0.0099** (↑ **0.28%**)	**0.8204±0.0100** (↑ **0.29%**)	**0.8131±0.0063** (↑ **1.02%**)	**0.8060±0.0089** (↑ **1.33%**)	**0.7973±0.0137** (↑ **2.36%**)
	MidGCN	0.8340±0.0021	0.8001±0.0037	0.7895±0.0037	0.7780±0.0055	0.7566±0.0057	0.7417±0.0072
	MidGCN-Our	**0.8349±0.0016** (↑ **0.09%**)	**0.8035±0.0044** (↑ **0.34%**)	**0.7943±0.0047** (↑ **0.48%**)	**0.7829±0.0074** (↑ **0.49%**)	**0.7645±0.0057** (↑ **0.79%**)	**0.7422±0.0085** (↑ **0.05%**)

Table 13: Accuracy of node classification on Cora under combination with Ours using Metattack

Data set	RAP / Mod.	0%	5%	10%	15%	20%	25%
Cora	GCN	0.8300±0.0100	0.7535±0.0184	0.6683±0.0340	0.5908±0.0408	0.5055±0.0330	0.4397±0.0358
	TopFeaRe	**0.8324±0.0101** (↑ **0.24%**)	**0.8032±0.0128** (↑ **4.97%**)	**0.7886±0.0131** (↑ **12.03%**)	**0.7807±0.0120** (↑ **18.99%**)	**0.7709±0.0126** (↑ **26.54%**)	**0.7521±0.0159** (↑ **31.24%**)
	GAT	0.8328±0.0101	0.7965±0.0175	0.7784±0.0190	0.7136±0.0463	0.6655±0.0390	0.6178±0.0441
	GAT-Our	**0.8388±0.0110** (↑ **0.60%**)	**0.8122±0.0122** (↑ **1.57%**)	**0.8019±0.0121** (↑ **2.35%**)	**0.7974±0.0109** (↑ **8.38%**)	**0.7790±0.0126** (↑ **11.35%**)	**0.7704±0.0155** (↑ **15.26%**)
	HANG	0.8192±0.0092	0.7894±0.0137	0.7724±0.0185	0.7419±0.0256	0.7154±0.0245	0.6946±0.0438
	HANG-Our	**0.8346±0.0090** (↑ **1.54%**)	**0.8036±0.0117** (↑ **1.42%**)	**0.7867±0.0145** (↑ **1.43%**)	**0.7852±0.0110** (↑ **4.33%**)	**0.7731±0.0132** (↑ **5.77%**)	**0.7671±0.0190** (↑ **7.25%**)
	MidGCN	0.8310±0.0041	0.7581±0.1582	0.7888±0.0093	0.7543±0.0111	0.7550±0.0135	0.7443±0.0083
	MidGCN-Our	**0.8323±0.0036** (↑ **0.13%**)	**0.8148±0.0037** (↑ **5.67%**)	**0.7944±0.0078** (↑ **0.56%**)	**0.7641±0.0118** (↑ **0.98%**)	**0.7645±0.0049** (↑ **0.95%**)	**0.7469±0.0121** (↑ **0.26%**)

Table 14: Accuracy of node classification on Amazon Photo under combination with Ours using Metattack

Data set	RAP / Mod.	0%	5%	10%	15%	20%	25%
Amazon Photo	GCN	0.9215±0.0034	0.8242±0.0774	0.8281±0.0581	0.7529±0.1425	0.6295±0.1496	0.6108±0.1526
	TopFeaRe	**0.9243±0.0118** (↑ **0.28%**)	**0.8486±0.0682** (↑ **2.44%**)	**0.8532±0.0338** (↑ **2.51%**)	**0.7754±0.1088** (↑ **2.25%**)	**0.7108±0.1246** (↑ **8.13%**)	**0.6847±0.1278** (↑ **7.39%**)
	GAT	0.9354±0.0018	0.9173±0.0039	0.7824±0.1926	0.7239±0.1575	0.6528±0.1737	0.5811±0.1824
	GAT-Our	**0.9448±0.0018** (↑ **0.94%**)	**0.9257±0.0030** (↑ **0.84%**)	**0.8598±0.1236** (↑ **7.74%**)	**0.8079±0.1337** (↑ **8.40%**)	**0.7731±0.1729** (↑ **12.03%**)	**0.7614±0.1384** (↑ **18.03%**)
	HANG	0.9347±0.0032	0.9162±0.0054	0.9151±0.0039	0.9093±0.0068	0.9016±0.0056	0.8992±0.0061
	HANG-Our	**0.9441±0.0025** (↑ **0.94%**)	**0.9245±0.0045** (↑ **0.83%**)	**0.9236±0.0033** (↑ **0.85%**)	**0.9203±0.0058** (↑ **1.10%**)	**0.9184±0.0048** (↑ **1.68%**)	**0.9052±0.0052** (↑ **0.60%**)
	MidGCN	0.7725±0.0025	0.7439±0.0049	0.6595±0.0082	0.5708±0.0170	0.4875±0.0088	0.4675±0.0113
	MidGCN-Our	**0.7903±0.0053** (↑ **1.78%**)	**0.7473±0.0064** (↑ **0.34%**)	**0.6651±0.0067** (↑ **0.56%**)	**0.5969±0.0102** (↑ **2.61%**)	**0.5020±0.0088** (↑ **1.45%**)	**0.4759±0.0101** (↑ **0.84%**)

Table 15: Accuracy of node classification on PubMed under combination with Ours using Metattack

Data set	RAP / Mod.	0%	5%	10%	15%	20%	25%
PubMed	GCN	0.8340±0.0021	0.8001±0.0037	0.7895±0.0037	0.7780±0.0055	0.7566±0.0057	0.7417±0.0072
	TopFeaRe	**0.8824±0.0054** (↑ **4.84%**)	**0.8672±0.0054** (↑ **6.71%**)	**0.8579±0.0095** (↑ **6.84%**)	**0.8514±0.0110** (↑ **7.34%**)	**0.8365±0.0083** (↑ **7.99%**)	**0.8177±0.0038** (↑ **7.60%**)
	GAT	0.8583±0.0012	0.8271±0.0032	0.6808±0.0236	0.5419±0.0128	0.3640±0.0416	0.3165±0.0394
	GAT-Our	**0.8640±0.0004** (↑ **0.57%**)	**0.8348±0.0061** (↑ **0.77%**)	**0.7923±0.0140** (↑ **11.15%**)	**0.6546±0.0271** (↑ **11.27%**)	**0.6070±0.0091** (↑ **24.30%**)	**0.5848±0.0220** (↑ **26.83%**)
	HANG	0.8712±0.0006	0.8001±0.0037	0.7895±0.0037	0.7780±0.0055	0.7566±0.0057	0.7417±0.0072
	HANG-Our	**0.8711±0.0008** (↓ **0.01%**)	**0.8665±0.0027** (↑ **6.64%**)	**0.8637±0.0014** (↑ **7.42%**)	**0.8571±0.0042** (↑ **7.91%**)	**0.8492±0.0023** (↑ **9.26%**)	**0.7940±0.0103** (↑ **5.23%**)
	MidGCN	0.8340±0.0021	0.8001±0.0037	0.7895±0.0037	0.7780±0.0055	0.7566±0.0057	0.7417±0.0072
	MidGCN-Our	**0.8581±0.0053** (↑ **2.41%**)	**0.8403±0.0062** (↑ **4.02%**)	**0.8111±0.0127** (↑ **2.16%**)	**0.7959±0.0219** (↑ **1.79%**)	**0.7784±0.0123** (↑ **2.18%**)	**0.7532±0.0308** (↑ **1.15%**)