Grimm: A Plug-and-Play Perturbation Rectifier for Graph Neural Networks Defending Against Poisoning Attacks

Ao Liu[1], Wenshan Li[2*], Beibei Li[1], Wengang Ma[1], Tao Li[1], Pan Zhou[3]

[1]School of Cyber Science and Engineering, Sichuan University
[2]School of Cyber Science and Engineering, Chengdu University of Information Technology
[3]School of Cyber Science and Engineering, Huazhong University of Science and Technology
{aliu, libeibei, mawengang941206, litao}@scu.edu.cn, helenali@cuit.edu.cn, panzhou@hust.edu.cn

Abstract

Recent studies have revealed the vulnerability of graph neural networks (GNNs) to adversarial poisoning attacks on node classification tasks. Current defensive methods require substituting the original GNNs with defense models, regardless of the original's type. This approach, while targeting adversarial robustness, compromises the enhancements developed in prior research to boost GNNs' practical performance. Here we introduce GRIMM, the first plug-and-play defense model. With just a minimal interface requirement for extracting features from any layer of the protected GNNs, GRIMM is thus enabled to seamlessly rectify perturbations. Specifically, we utilize the feature trajectories (FTs) generated by GNNs, as they evolve through epochs, to reflect the training status of the networks. We then theoretically prove that the FTs of victim nodes will inevitably exhibit discriminable anomalies. Consequently, inspired by the natural parallelism between the biological nervous and immune systems, we construct GRIMM, a comprehensive artificial immune system for GNNs. GRIMM not only detects abnormal FTs and rectifies adversarial edges during training but also operates efficiently in parallel, thereby mirroring the concurrent functionalities of its biological counterparts. We experimentally confirm that GRIMM offers four empirically validated advantages: 1) *Harmlessness*, as it does not actively interfere with GNN training; 2) *Parallelism*, ensuring monitoring, detection, and rectification functions operate independently of the GNN training process; 3) *Generalizability*, demonstrating compatibility with mainstream GNNs such as GCN, GAT, and GraphSAGE; and 4) *Transferability*, as the detectors for abnormal FTs can be efficiently transferred across different systems for one-step rectification.

Extended version — https://arxiv.org/abs/2412.08555

1 Introduction

Graph neural networks (GNNs), benefitting from the message passing (MP) strategy, have achieved remarkable success in node classification tasks (Wu et al. 2021). However, GNNs are easily poisoned by imperceptible adversarial perturbations (i.e., inserted/deleted edges) on graph structure during their training phase (Li et al. 2021; Zheng et al. 2021; Wang and Gong 2019; Xi et al. 2021; Meng et al. 2023; Wu et al. 2022). Even slight but deliberate perturbations [1] introduced to the training set can poison the target GNN, and then rewire the MP pattern driven by the poisoned GNN, to further misclassify nodes to the target categories (Dai et al. 2018; Zügner, Akbarnejad, and Günnemann 2018). This may lead to critical issues in many application areas (Che et al. 2020; Chaturvedi and Garain 2021; Katzir and Elovici 2021; Liu, Akhtar, and Mian 2020), including those where perturbations undermine public trust (Kreps and Kriner 2020), interfere with human decision making (Walt, Jack, and Christof 2019), and affect human health and livelihoods (Samuel et al. 2019). More seriously, it's proved that non-robust GNNs are inevitably poisoned once adversaries take them as the target (Liu et al. 2022).

A plethora of robustness models aim to enhance protected GNNs with add-on functions to defend against edge-perturbing attacks, which causes the unavoidable dismissal of the original inner function of the non-robustness GNNs. As representative examples: (1) RGCN (Zhu et al. 2019) replaces the hidden representations of nodes in each graph convolutional network (GCN) (Kipf and Welling 2017) layer to the Gaussian distributions, to further absorb the effects of adversarial changes. (2) GCN-SVD (Entezari et al. 2020) combines a singular value decomposition (SVD) filter prior to GCN to eliminate adversarial edges in the training set. (3) STABLE (Li et al. 2022) reforms the forward propagation of GCN by adding functions that randomly recover the roughly removed edges. (4) EGNN (Liu et al. 2021) leverages graph smoothing techniques based on transductive model PPNP & APPNP (Gasteiger, Bojchevski, and Günnemann 2018), to confine the permutation setting space, effectively excluding the majority of non-smooth permutations. (5) GRN (Liu et al. 2024) incorporates local signals into the central node's representation to improve resonance-based robustness.

Unfortunately, deploying these defense models requires (partially or completely) replacing the original GNN, inevitably leading to the loss of its custom functions and features. In practical applications, diverse GNNs, such as GCN, graph attention network (GAT) (Veličković et al. 2017), and GraphSAGE (Hamilton, Ying, and Leskovec 2017), are de-

*Corresponding author

[1]We prioritize defense against graph structure attacks due to their heightened destructiveness compared to feature attacks.

signed for various tasks according to differentiated requirements. For instance, a node classification task in large graphs relies on inductive frameworks such as GraphSAGE, instead of transductive frameworks such as GCN. However, GCN-based defending models discards the generalizability of inductive frameworks, leading to excessive time spend. This approach sacrifices the efforts made by previous research to enhance GNNs' practical performance.

The examination of the issue necessitates the integration of a non-disruptive, plug-and-play defensive mechanism. The core challenge lies in identifying a describable anomaly pattern that is directly related to the data and independent of the model architecture.

For this challenge, we theoretically demonstrate that the manifestation of anomalies in the MP of a compromised GNN is discernible. This is substantiated by the observable differences in the *feature trajectories* (FTs) of attacked versus non-attacked nodes throughout the training process. These trajectories provide a detectable interface for the implementation of external defensive strategies.

However, formulating anomalous trajectories of node features is possible, yet capturing these observable antibody behaviors remains challenging attributes to two primary issues: 1) Abnormal trajectories are aggregated by illegal messages that are hide inside GNNs and pass along the edges of the graph throughout the training epochs. Due to the non-Euclidean nature of graphs, monitoring fine-grained (including node- and edge-grained) MP and converting them into computable Euclidean tensors is almost intractable, much less capturing these illegal messages. This problem is even more pronounced in some transductive GNNs (such as GCN). 2) Sufficient samples of illegal messages can only be obtained after the GNN is already poisoned, and these samples are non-transferable due to the randomness of the adversaries, resulting in the inability of the classifier to predict the universal pattern of illegal messages.

In addressing this challenge, we have identified a biologically inspired method. Our research indicates that mimicking the mechanism by which the human immune system (HIS) detects viruses (Kipnis 2016) serves as an efficacious strategy for the issues delineated previously. This efficacy is ascribed to the synergistic interaction between the HIS and the human nervous system (HNS). The HNS, serving as a biological prototype for neural networks (Morton, Schlichting, and Preston 2020), collaborates with the HIS to safeguard the human body against invasive viral antibodies (Brodin and Davis 2017), without inducing mutual harm (Kenney and Ganta 2014).

Therefore, it is intuitive that capturing the distributed illegal messages and rectify them in parallel by imitating the mechanism of HIS. Here we propose GRIMM which inherits the merits of the HIS: *Harmless*. GRIMM does not actively intervene with the inner function of the protected GNN. It is a plug-and-play system in practical applications. *Parallel*. GRIMM detects adversarial edges and rectifies the perturbed graph parallel with the protected GNN during its training phase. *Generalizable*. GRIMM can cooperate with the mainstream GNNs such as GCN, GAT and GraphSAGE while fully maintaining their original inner functions. *Transferable*. GRIMM can improve its defending ability by inter-system information from another system.

In practical application, initially, GRIMM is integrated into any layer of the protected GNNs before training begins. It computes the feature trajectories (FTs) of nodes and edges in real-time from this layer's outputs, transforming global message passing in non-Euclidean geometry into computable tensors within Euclidean space. This enables global monitoring of the target. GRIMM then starts training simultaneously with the target GNN. Given the hyperparameters of the target GNN, GRIMM systematically generates potential detectors from feasible FTs. It evolves these FTs into detectors by selecting a subset of reliable FTs and inversely detecting anomalies within the target. In the final training phase, these detectors identify adversarial edges and correct perturbations in the database in real-time. This process runs concurrently with GNN training, ensuring a clean dataset and a well-trained GNN at completion.

Our contributions are:

- To the best of our knowledge, we propose the first plug-and-play defense model against poisoning attacks.
- We theoretically demonstrate that nodes under attack form discriminable FTs.
- We reconstruct the implementation of the HIS specifically for graph learning scenarios.
- We systematically evaluate our proposed defense model on real-world datasets.

2 Preliminaries

Message Passing GNN We consider connected graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting $N = |\mathcal{V}|$ nodes, where $\mathcal{E}$ is the set of edges. Let $\mathbf{A} \in \{0,1\}^{N\times N}$ be the adjacency matrix. Let generic symbol $\mathbf{L}$ be the Laplacian in its broadest sense. Denote the GNN's number of layers as L and the feature dimension of its ℓ^{th} layer output as d_ℓ. The feature and one-hot label matrix are $\mathbf{Z} \in \mathbb{R}^{N\times d_0}$ and $\mathbf{Y} \in \mathbb{R}^{N\times d_L}$ respectively. The edge connected nodes v_i and v_j is (v_i, v_j) or (v_j, v_i). The neighborhood $\mathcal{N}_i$ of node v_i consists of all nodes v_j for which $(v_i, v_j) \in \mathcal{E}$. Let $\deg_i$ be the degree of node v_i. The feature vector and one-hot label of node v_i are $\mathbf{z}_i$ and $\mathbf{y}_i$.

Feature Trajectories (FTs) Formed by MP GNNs are vulnerable to poisoning attacks that subtly introduce perturbations, manifesting progressively across training epochs. This gradual emergence suggests that oscillatory trends in node signals across GNN layers might reveal latent adversarial edges. Our analysis concentrates on the evolution of output features at layer ℓ, represented by $\mathbf{z}_{i,\ell}$ for each node i in $\mathcal{V}$. To capture this evolution, we incorporate a temporal dimension, denoting the feature of node i at the t-th epoch at layer ℓ as $\mathbf{z}_{i,\ell}^{(t)}$ and the collective features at this layer as $\mathbf{Z}_\ell^{(t)}$. Initially, $\mathbf{z}_{\cdot,\ell}^{(0)}$ and $\mathbf{Z}_\ell^{(0)}$ capture the baseline state of features. The trajectory of $\mathbf{z}_{i,\ell}^{(t)}$ in $\mathbb{R}^{d_\ell}$ is described by:

$$\mathcal{T}_{i,\ell} = [\mathbf{z}_{i,\ell}^{(0)}, \mathbf{z}_{i,\ell}^{(1)}, \ldots]. \tag{1}$$

Section 3.2 discusses how deviations in $\mathcal{T}_{i,\ell}$ indicate attacks, aiding in corrective measures.

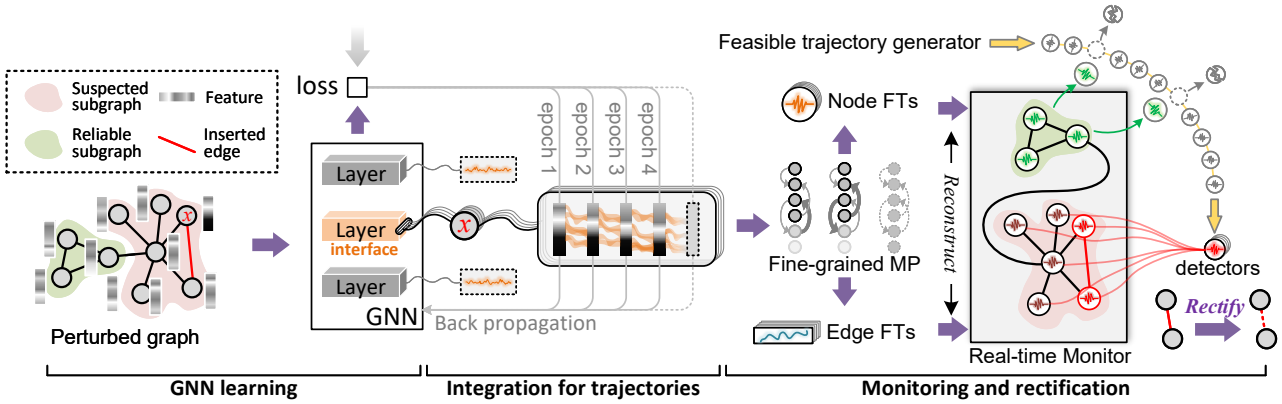

Figure 1: The general workflow of GRIMM.

Feature aggregation through edges is pivotal in GNN learning. During the message passing from epoch t to $t+1$, the feature quantity transmitted between nodes i and j affects $\mathbf{z}_{i,\ell}^{(t)}$ and $\mathbf{z}_{j,\ell}^{(t)}$, although each node's feature also reflects contributions from all adjacent edges. For edge (i,j), the features transmitted at epoch t at layer ℓ are denoted as $\mathbf{z}_{(i,j),\ell}^{(t)}$ or $\mathbf{z}_{(j,i),\ell}^{(t)}$, indicating bidirectional feature flow in undirected graphs. The trajectory of features on the edge (i,j) at layer ℓ evolves as: $\mathcal{T}_{(i,j),\ell} = [\mathbf{z}_{(i,j),\ell}^{(0)}, \mathbf{z}_{(i,j),\ell}^{(1)}, \ldots]$. The method for obtaining FTs is detailed in the Appendix C.

Mechanism of Identifying Viruses in the HIS The mechanism for virus identification in the HIS is efficiently replicated in the domain of artificial immune systems (AIS) (Dasgupta 2006) for viruses detection. The core processes of AIS and the corresponding elements in GRIMM (indicated within parentheses) are: 1) Creation of Immune Cells (**feasible FTs**): Immune cells are generated indiscriminately to counter all possible antigens. 2) Elimination of Redundant Cells (**reliable FTs**): Immune cells responsive to benign antigens are eradicated. 3) Virus Detection: Active immune cells (**detectors**) undertake the detection of viruses. 4) Vaccine (**transferred detectors**) Production: Functional immune cells synthesize vaccines, which are then disseminated across various human immune systems for broader protection. *Note that AIS primarily as a conceptual tool to integrate HIS strategies rather than as a direct implementation method.*

3 The Proposed Model

3.1 Overview

The primary workflow of GRIMM encompasses 3 stages:

► **Integrate FTs**: This involves interfacing with a certain layer of the protected GNN to monitor the feature trajectories of all nodes and edges at that layer. Specifically, for a given layer ℓ and nodes i, j, it aims to acquire real-time updated $\mathcal{T}_{i,\ell}$ and $\mathcal{T}_{(i,j),\ell}$.

► **Detect abnormal FTs**: This phase is dedicated to identifying abnormal edge FTs, formalized as finding a classifier:

$$\mathcal{I}_{edge} : \{\mathcal{T}_{(i,j),\ell} : \forall (i,j) \in \mathcal{E}\} \rightarrow \{\text{normal}, \text{abnormal}\} \quad (2)$$

Edges classified as "abnormal" are suspected of attacks.

► **Rectify perturbations**: Post the detection of abnormalities, the identified perturbations are rectified upon, informing further rectifications.

Motivation During the training phase of GNN, node representations dynamically evolve, forming trajectories in an equidimensional feature space. We hypothesize that these trajectories can reveal adversarial entities through the observed behaviors of antibodies. To investigate, we replicate a poisoning attack on a GNN, employing the Metattack methodology (Zügner and Günnemann 2018) on a 4-layer GCN with the Cora dataset. We introduce a frozen decoder (mapping from $\mathbb{R}^{d_i}$ to $\mathbb{R}$) to the penultimate hidden layer of the GCN to capture one-dimensional features of this layer.

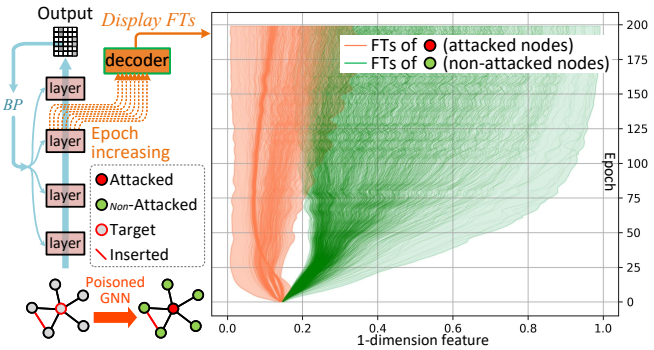

Figure 2: FTs of attacked vs. non-attacked nodes.

The decoder distinguishes the trajectories of nodes under attack (maliciously misclassified) from those not targeted (correctly classified) during the GCN's training. These trajectories, along with the methodology and results, are presented in Figure 2. It is evident that adversarial edges induce distinctive trajectory patterns in their adjacent nodes, unlike non-attacked nodes. This observation underscores the potential of AIS in detecting and intercepting these illicit communications within the network.

3.2 Theoretical Foundations

FTs' Discriminability We first fortify the observations noted in Section 3.1 (Motivation) with a theoretical foundation through the following theorem:

Theorem 1: Consider a GNN undergoing a poisoning attack. Let $\mathcal{V}_{adv}$ and $\mathcal{V}_{non}$ respectively denote the sets of nodes with categories that have been compromised and those that remain uncompromised, The following classification function exists for all layer ℓ:

$$\mathcal{I}_{node} : \mathcal{T}_{i,\ell} \to \{\mathcal{T}_{j,\ell}, \mathcal{T}_{k,\ell}\}, \text{ s.t.: } \forall i \in \mathcal{V}, \forall j \in \mathcal{V}_{adv}, \forall k \in \mathcal{V}_{non}. \tag{3}$$

That is, FTs of attacked and non-attacked nodes are discriminable.

Proof in Appendix D. This conclusion unveils an inevitable consequence of the attack: once an attack occurs, the FTs inevitably reveal the malicious activity. This finding lays the theoretical groundwork for detecting abnormal FTs.

Lower bound of FTs' inner products Given the operation of the MP process under $\mathcal{M}(\cdot)$ with predefined hyperparameters such as the learning rate η, the fluctuation in node signal magnitudes is inherently constrained. This constraint ensures predictability in the oscillation range of node signals across $\mathcal{E}$, regardless of edge distribution.

Drawing on insights from Theorem 1, we observe that signal trajectories tend to stabilize during training. This stability is demonstrated by the limited spatial angle between direction vectors across successive epochs, implying that the inner product of these vectors meets a minimum threshold. Specifically, for node i, consider:

$$\begin{aligned} \gamma_{i,\ell}^{\text{node}} &= (\mathbf{z}_{i,\ell}^{(t+1)} - \mathbf{z}_{i,\ell}^{(t)}) \cdot (\mathbf{z}_{i,\ell}^{(t+2)} - \mathbf{z}_{i,\ell}^{(t-1)}), \\ \gamma_{(i,\cdot),\ell}^{\text{edge}} &= (\mathbf{z}_{(i,\cdot),\ell}^{(t+1)} - \mathbf{z}_{(i,\cdot),\ell}^{(t)}) \cdot (\mathbf{z}_{(i,\cdot),\ell}^{(t+2)} - \mathbf{z}_{(i,\cdot),\ell}^{(t-1)}), \end{aligned} \tag{4}$$

we have the following proposition.

Proposition 1: Let $\mathcal{P}(\mathbf{Z}_{\ell-1}; \mathcal{E}) = \mathbf{L}(\mathbf{L}\mathbf{Z}_{\ell-1}\mathbf{Z}_{\ell-1}^{\top})^{\top})^{\top}$ be a modular MP model, $\mathbf{J}$ be ones matrix, and $(\cdot)^{\circ-1}$ be Hadamard inverse. At the end of the t^{th} epoch, for any node i and layer ℓ, we have

$$\begin{aligned} &\gamma_{i,\ell}^{\text{node}} \geq \lambda \quad \text{and} \quad \gamma_{(i,\cdot),\ell}^{\text{edge}} \geq \lambda, \text{s.t.,} \\ &\lambda = \eta^2 \max_k \|((\mathcal{P}(\mathbf{Z}_{\ell-1}; \mathcal{E}))((\mathbf{J} - \mathbf{L}\mathbf{Z}_{\ell-1}\mathbf{W}_{\ell}^{(t)})^{\circ-1} - \mathbf{Y}))_{k,\cdot}\|_2^2, \end{aligned} \tag{5}$$

Proof in Appendix E. Note that we only gives the mathematical bound for GCN since its a non-intuitive transductive message passing model. Bound of GAT and GraphSAGE are assigned as a empirical constant.

3.3 Detecting Abnormal FTs and Rectifying Perturbations

The essence of detecting abnormal FTs lies in obtaining detectors for these abnormal FTs based solely on reliable FTs. This process unfolds in two steps:

1) Under the constraints set forth in Proposition 1, exhaustively enumerate all feasible FTs. 2) Eliminate those FTs that exhibit a relatively small mean squared error (MSE) in comparison to the reliable FTs. The remaining entities constitute effective detectors.

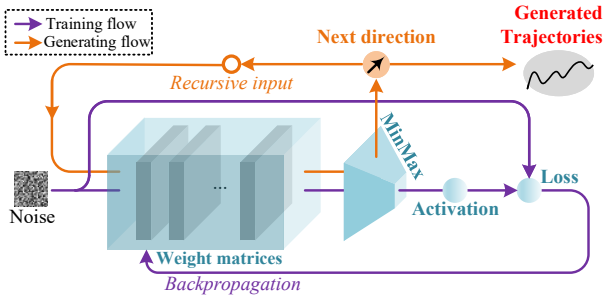

Figure 3: The general workflow of the trajectory generator.

A Cyclic Self-Supervised Generator of FTs Feasible trajectories are constrained by Proposition 1. The FT generator creates valid trajectories. However, pre-training trajectory generation is impractical due to dynamic adversarial intervention in MP of GNNs. Thus, synchronous real-time generation is essential as training epochs progress, as illustrated in Figure 3.

Figure3 outlines two processes: training and generation, indicating the generator's self-training and cyclic real-time direction vector output based on the previous epoch's vector. Specifically, for a valid vector v, at layer p and epoch e, the optimal generator $G(\cdot;\theta)$ with trainable weights θ executes $G(v;\theta) \cdot v \leq \gamma_i^{\text{node}}$. This requires first training $G(\cdot)$ by addressing the self-supervised loss derived from Proposition1, i.e., $\min_\theta \left(\text{Sigmoid}(G(noise;\theta) \cdot noise) - \gamma_i^{\text{node}}\right)$, where $noise$ is randomly sampled noise. Then, for a target trajectory length ϱ, the trained generator yields a vector set:

$$[\mathcal{T}^{\text{init}}, G(\mathcal{T}^{\text{init}};\theta), G(G(\mathcal{T}^{\text{init}};\theta);\theta), \ldots], \tag{6}$$

where $\mathcal{T}^{\text{init}}$ is the initial direction vector. Finally, detectors are reconstituted by generated vector sets. The generator continues to produce feasible trajectories until sufficient, i.e., they almost span the entire feasible domain of FTs.

Producing detectors Using Negative Selection Algorithm (NSA) The NSA posits that the elimination of normal samples from all feasible samples results in a detector $\mathcal{T}_{det}$ for abnormal samples. We have demonstrated the feasibility of this approach for FT samples in Section 3.2 (Theoretical Foundations). In other words, by simply discarding those feasible FTs generated by the generator, which have a MSE with reliable FTs less than a given threshold ρ what remains are effective detectors for abnormal FTs. This method is straightforward yet intuitive. In the training phase of GNNs, the dimensionality of FTs increases as the number of training epochs t increases. Hence, we define checkpoints; detectors are generated when the epoch reaches this point. Between two checkpoints, we only monitor the GNN without taking specific actions, until the epoch reaches the next checkpoint. The set of detector is denoted as $\mathbb{T}$.

Detecting **Detecting abnormal FTs**. The determination of whether a FT is abnormal is contingent upon its average MSE in relation to all detectors. If this average MSE falls below the threshold ρ, then the FT in question is classified as

an abnormal FT. In other words, given the layer ℓ, the function $\mathcal{I}_{node}$, as delineated in Theorem 1, can be instantiated as

$$\forall i \in \mathcal{V}, \mathcal{T}_{det} \in \mathbb{T}, \mathcal{I}_{node}(\mathcal{T}_{i,\ell}) = \begin{cases} \mathcal{T}_{adv}, & MSE(\mathcal{T}_{i,\ell}, \mathcal{T}_{det}) \leq \rho \\ \mathcal{T}_{non}, & \text{otherwise}, \end{cases} \tag{7}$$

where $\mathcal{T}_{adv}$ and $\mathcal{T}_{non}$ are FTs formed on the attacked and non-attacked nodes. **Detecting inserted edges**. As adversarial edges pass illegal messages which may mislead the classification of the target nodes, the node FTs on their directly connected node will act abnormally. Therefore, we can identify inserted edges according to the abnormal node FTs. As shown in Figure 4.(a), for each $\mathcal{T}_i$, we query the trajectories $\mathcal{T}_{i,(i,j)}$ of edges which directly connect to node i where $(i,j) \in \mathcal{E}$. If any abnormal trajectories $\mathcal{T}_{i,(i,k)}$ exist, the edge (i,k) is detected as the inserted edge. **Detecting deleted edges**. Similar to the aforementioned analysis, if all trajectories of edges connect to node i are normal, while the FT of node i is abnormal, we can confirm that one of the edges should be connected to node i is deleted. Then, as shown in Figure 4.(b), aiming at locating the deleted edge (i,o), $\mathcal{G}'$ can be rectified circularly until the trajectory on (i,o) is identified as normal FT, i.e., if $\mathcal{T}_{i,(i,o)}$ is abnormal, edge (i,o) is detected as the deleted edge.

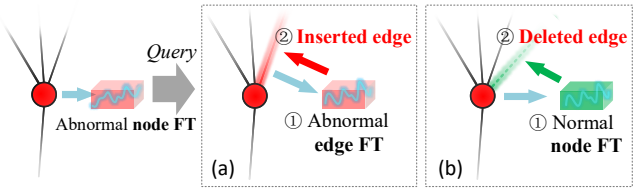

Figure 4: The detection method for adversarial edges.

Rectifying perturbations Once the adversarial edge is detected, GRIMM rectifies the perturbed graph and meanwhile does not terminate the training process. However, the model is still poisoned to a certain extent despite some adversarial edges on the graph are rectified immediately. Therefore, after rectifying, we rollback the version of the trainable matrix W to the previous δ epochs, to thus reduce the harm caused by the rectified adversarial edges. For instance, if an edge is detected as the inserted edge in epoch t, the perturbed graph $\mathcal{G}'$ is rectified as $\mathcal{G}'_{r,t} = \{\mathbf{Z}, \mathcal{E}'_{r,t}\}$ by deleting the corresponding edge. In the experiments, we find that rolling back of $\mathbf{W}$ will cause drop in accuracy during the training phase. Then, the limited lower accuracy caused by the adversarial edges will be broken through after several epochs. Pseudo-code of GRIMM is provided in Appendix B.

4 Experiments

The effectiveness of GRIMM is evaluated under multiple aspects including: 1) global accuracy, 2) harmlessness, 3) transferability, 4) interface position, 5) sensitivity of MSE's threshold, and 6) runtime comparison of robust functions, and 7) internal rectification details (*c.f.* Appendix A).

Datasets. Our approaches are evaluated on six real-world datasets widely used for studying graph adversarial attacks (Liu et al. 2022, 2024). These datasets are two types: *Small graphs*, which include the citation datasets Cora and Citeseer, as well as Polblogs, representing social networking data. *Large graphs*, which include *Brain*(Wang et al. 2017), *Pubmed*, and *Reddit*(Hamilton, Ying, and Leskovec 2017). *For each dataset, we randomly partitioned 1/10th of the region to serve as a reliable subgraph*, ensuring the absence of perturbations within this designated area.

Baselines. The proposed GRIMM model not only protects non-defense GNNs against poisoning attacks but also surpasses other defense models. Baseline models are categorized into three groups:

Non-defense GNNs. To illustrate GRIMM's efficacy against edge-perturbing attacks, we selected representative GNNs: 1) *GCN*, a prevalent architecture, 2) *GAT*, a typical GNN variant, 3) *GraphSAGE*, effective in aggregating spatial features.

Comparison defending models. We benchmark GRIMM against defense models: 1) *RGCN*, using Gaussian distributions to mitigate adversarial impacts, 2) *GNN-SVD*, employing a truncated SVD for adjacency matrix approximation, 3) *Pro-GNN*, focusing on intrinsic node properties for robustness, 4) *Jaccard* (Wu et al. 2019), based on Jaccard similarity for defense, 5) *EGNN*, filtering perturbations via ℓ_1- and ℓ_2-based graph smoothing.

Attack methods. Experiments consider attack strategies: 1) *Metattack*, a meta-learning based approach, 2) *CLGA* (Sixiao et al. 2022), an unsupervised tactic, 3) *RL-S2V* (Dai et al. 2018), leveraging reinforcement learning.

All experiments were conducted using an NVIDIA RTX 4080s GPU, an Intel i7-14700-KF CPU, and 64GB of RAM.

Accuracy After Rectification Here we evaluate the global classification efficacy of GRIMM by applying Metattack to disrupt target models and measuring the post-training classification accuracy, with results documented in Table 1. The evaluation captures outcomes for models shielded by GRIMM after global rectification and once model loss stabilizes. The mean accuracy and its deviation are presented. Observations indicate that GRIMM effectively protects non-defense GNNs under attacks and outperforms leading robust GNNs, with few exceptions: 1) In the Pubmed dataset at $p_r = 10\%$, EGNN, using graph smoothing for enhanced adversarial robustness, handles localized perturbations effectively, but such scenarios are infrequent and GRIMM's accuracy improves as p_r increases. 2) For the Polblogs dataset at $p_r = 0\%$, GRIMM slightly lags behind Pro-GNN by 0.32%. However, as p_r increases, GRIMM-protected models show the smallest decline in accuracy among baselines, maintaining a leading position.

In non-adversarial settings ($p_r = 0\%$), rectification aims to boost accuracy by optionally modifying edges, as no changes are needed for a clean graph. When reliable FTs are unattainable due to the absence of a dependable subgraph on $\mathcal{G}'$, we base our experiments on reliable FTs recommended from an exogenously reliable subgraph perturbed at a 20% rate, achieving high accuracy.

Dataset	p_r	Unprotected models			Non-symbiotic defending models					Protected by GRIMM			Protected by GRIMM (E)		
		GCN	GAT	SAGE	RGCN	SVD	Pro	Jaccard	EGNN	GCN	GAT	SAGE	GCN	GAT	SAGE
Cora	20	59.02	59.13	63.91	59.01	56.37	64.38	73.24	69.52	73.49	73.84	**79.51**	71.02	65.70	75.40
Citeseer	20	62.73	60.14	68.81	62.90	57.65	55.54	66.92	65.61	70.76	72.42	**76.91**	68.42	66.01	75.48
Pubmed	20	70.02	69.89	72.09	70.88	81.54	82.57	76.61	78.91	75.10	**79.30**	74.05	73.39	78.04	76.89
Polblogs	20	52.33	50.28	54.12	58.04	55.41	73.60	70.55	75.95	**82.11**	81.02	78.62	76.47	76.44	75.40

Table 1: Classification accuracy (%) on the attacked graph after rectifying. GRIMM (E) means producing detectors based on the exogenous reliable FTs. SVD, Pro, and SAGE is the abbreviation of GNN-SVD, Pro- GNN and GraphSAGE.

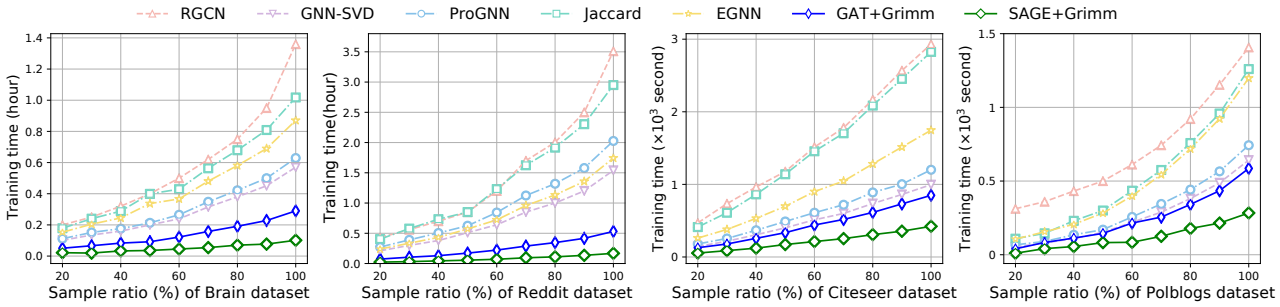

Figure 5: Training time under increased sample ratio.

Harmfulness of GRIMM and Other Models The impact of a defense model can be quantitatively assessed by its training duration, particularly when transductive models are used where inductive methods are preferable (e.g., large-scale graphs), leading to a significant increase in computational load. The efficiency of GRIMM is demonstrated by its relatively shorter training duration compared to other defense models. This was empirically tested across various attack conditions, documenting the training times until optimal performance was achieved, indicated by loss convergence. The additional time burden imposed by GRIMM, expressed as a percentage increase over the original training time, is detailed in Tables 2 and 4 (*c.f.* Appendix A). On smaller graphs, traditional defense models can extend training up to 3-5 times longer than usual. In contrast, GRIMM introduces minimal additional time, a benefit that is even more evident in larger datasets. For example, on the Reddit dataset, existing models like EGNN can increase training times by up to 18 times, whereas GRIMM maintains negligible extra time.

Real-world graphs often grow dynamically, making the ability to learn from scale-increasing graphs crucial for GNNs. To highlight GRIMM's efficiency, we conducted experiments on progressively larger samples of the Brain and Reddit datasets, measuring the training time of various defense models. The results, displayed in Figure 5, show that while traditional defense methods significantly extend training times as datasets expand, GRIMM ensures minimal increase, maintaining shorter overall durations across various scales, demonstrating its harmless integration with the protected model.

Transferability of Detectors GRIMM can rectify an attacked graph based on detectors transferred from another system, thus significantly reducing the time cost of graph rectification. In this section, we report the transferability of the detectors. We producedetectors by adopting Metattack on GCN, and transfer them to another GRIMM which is protecting different GNN models against different attacks. The perturbation rate of Metattack is set to 20% and the augmentation rate of CLGA is set to 10%. The transferred detectors are produced based on a GCN model and Cora dataset under the corresponding attack. The experimental results are shown in Table 3. The observation highlights that without any training, based on the detectors transferred from an external source, GRIMM can still detect and effectively rectify perturbations.

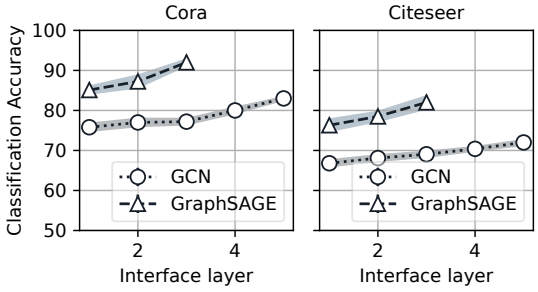

Figure 6: Classification accuracy of GRIMM after interfacing to the different layers.

Interface Position for GRIMM The primary experiments presented in the main text involve GRIMM interfacing with the penultimate layer of the protected GNNs. This section explores how the interface position within the GNNs affects global classification accuracy. We illustrate the mean value (central line) and standard deviation (shadow areas)

Dataset	Attack	Unprotected models			Defending models						Models protected by GRIMM		
		GCN	GAT	SAGE	RGCN	SVD	Pro	Jaccard	EGNN	*Avg. e. t.*	GCN	GAT	SAGE
Brain	Metattack	0.34	0.26	0.08	1.36	0.57	0.63	1.02	0.87	+982.7%	0.38 [+11.3%]	0.29 [+8.4%]	**0.10** [+25.4%]
	CLGA	0.35	0.26	0.08	1.39	0.59	0.62	1.06	0.86	+1009.8%	0.39 [+10.8%]	0.29 [+10.4%]	**0.11** [+36.9%]
	RL-S2V	0.35	0.27	0.08	1.38	0.58	0.64	1.06	0.87	+1017.2%	0.38 [+8.1%]	0.30 [+11.8%]	**0.10** [+28.1%]
Pubmed	Metattack	0.53	0.31	0.10	1.73	0.59	0.66	1.26	1.17	+1011.3%	0.57 [+7.7%]	0.34 [+10.4%]	**0.12** [+27.7%]
	CLGA	0.54	0.30	0.09	1.72	0.61	0.67	1.28	1.22	+1063.7%	0.59 [+8.5%]	0.35 [+15.0%]	**0.14** [+47.3%]
	RL-S2V	0.53	0.32	0.09	1.72	0.61	0.68	1.25	1.18	+1071.3%	0.58 [+8.9%]	0.36 [+14.4%]	**0.13** [+38.5%]
Reddit	Metattack	1.28	0.50	0.13	3.51	1.54	2.03	2.95	1.74	+1732.6%	1.34 [+4.6%]	0.53 [+6.7%]	**0.17** [+30.3%]
	CLGA	1.27	0.50	0.12	\	1.52	\	\	1.69	+1192.4%	1.32 [+3.9%]	0.54 [+7.4%]	**0.17** [+33.8%]
	RL-S2V	1.29	0.50	0.13	\	1.50	\	\	1.70	+1167.7%	1.34 [+3.6%]	0.54 [+7.9%]	**0.17** [+35.8%]

Table 2: Training time (hour) of defending models on large graphs. [+5%] indicating a 5% increase in training duration relative to the protected GNN, and *Avg.e.t.* represents the mean additional runtime in comparison to GraphSAGE.

Attack	Dataset	Cora		Citeseer		Pubmed	
	Target	GAT	SAGE	GAT	SAGE	GAT	SAGE
Meta	Before	59.20	62.21	61.08	68.29	69.51	71.77
	After	68.71	73.94	70.32	77.69	80.37	82.64
CLGA	Before	66.39	68.27	65.40	64.93	72.58	70.19
	After	76.34	80.85	78.42	77.39	79.65	78.52

Table 3: Classification accuracy (%) of poisoned GNNs (before) and GRIMM-protected GNNs (after). GRIMM rectifies perturbed graph based on transferred detectors.

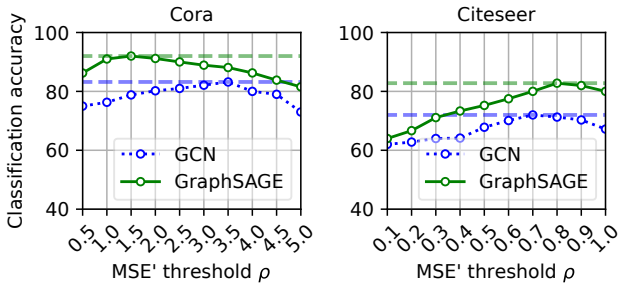

Figure 7: Sensibility of the threshold ρ for the affinity.

of results, repeated 10 times, in Figure 6. Figure 6 shows a clear trend: classification accuracy progressively improves as the interfaced layer's depth within the GNNs increases. This pattern aligns with the operational dynamics of GNNs, where features are extracted sequentially from graph data across layers. Thus, interfacing deeper within the network allows for the extraction of increasingly precise and sophisticated features. This enhancement in feature representation intrinsically boosts global classification accuracy, confirming the trend that deeper interfacing layers within the GNNs lead to improved performance.

Sensitivity of MSE's threshold ρ The process of identifying abnormal FTs in GRIMM depends on calculating their MSE) relative to the detectors. The choice of the threshold ρ, which sets the MSE boundary for detecting abnormalities, is crucial to GRIMM's performance dynamics. We then focus how accuracy responds to changes in ρ settings. The results of this investigation are visually presented in Figure 7. The graph shows that the permissible range for ρ is relatively wide, indicating that GRIMM has low sensitivity to fluctuations in ρ. This trait suggests that GRIMM's performance remains robust and stable, even with varying ρ values. Such resilience to the MSE threshold not only highlights the robustness of the GRIMM framework but also provides operational flexibility, allowing for some latitude in selecting ρ without significantly affecting the accuracy of abnormal FT detection.

Observations and Discussions The findings are summarized as follows:

1) *Vulnerability of GNNs*: Data from Table 1 highlights the vulnerability of unprotected GNNs (GCN, GAT, GraphSAGE) to perturbations, underscoring the need for robust defense mechanisms. 2) GRIMM*'s Superiority*: GRIMM outperforms other defense models in mitigating graph perturbations, although minor accuracy reductions are noted in some cases (Table 1). 3) *Performance in Non-adversarial Contexts*: In scenarios without adversarial attacks ($P_rate = 0\%$), GRIMM improves GNN structure and performance through the strategic addition of auxiliary edges (Table 1). 4) *Computational Efficiency*: GRIMM significantly reduces operational time compared to baseline models, especially noticeable in larger datasets (Table 2, Figure 5). 5) *Utility of Transferred Detectors*: GRIMM effectively employs transferred detectors to rectify perturbed graphs, avoiding the extensive training of new detectors and thereby saving computational resources (Table 3).

5 Conclusion

This paper presents the first plug-and-play defense model GRIMM against poisoning attacks for GNNs. GRIMM seamlessly integrates with various GNNs without disrupting their intrinsic functions, offering a parallel, non-intrusive, and generalizable defense mechanism. Underlying implementations of HIS are migrated to GNN. GRIMM can continuously monitor the MP of GNNs, detect adversarial edges and reflect the perturbed graph. Experiments demonstrated GRIMM protects mainstream GNNs including GCN, GAT and GraphSAGE from the most powerful attacks while outperforming the state-of-the-art defenses.

Acknowledgments

This work is supported in part by the National Key R&D Program of China (No. 2020YFB1805400), the National Science Foundation of China (No. U22B2027, 62032002, 62372313, 62172297, 62476107), the China Postdoctoral Science Foundation (No. 2024M752211), the Sichuan Science and Technology Program (No. 24QYCX0417), and the Youth Science Foundation of Sichuan (No. 25QNJJ4560).

References

Brodin, P.; and Davis, M. M. 2017. Human immune system variation. *Nature Reviews Immunology*, 17(1): 21–29.

Chaturvedi, A.; and Garain, U. 2021. Mimic and Fool: A Task-Agnostic Adversarial Attack. *IEEE TNNLS*, 32(4): 1801–1808.

Che, Z.; Borji, A.; Zhai, G.; Ling, S.; Li, J.; Min, X.; Guo, G.; and Callet, P. L. 2020. SMGEA: A New Ensemble Adversarial Attack Powered by Long-Term Gradient Memories. *IEEE TNNLS*, early access.

Dai, H.; Li, H.; Tian, T.; Huang, X.; Wang, L.; Zhu, J.; and Song, L. 2018. Adversarial attack on graph structured data. In *ICML*, 1115–1124.

Dasgupta, D. 2006. Advances in artificial immune systems. *IEEE Computational Intelligence Magazine*, 1(4): 40–49.

Entezari, N.; Al-Sayouri, S. A.; Darvishzadeh, A.; and Papalexakis, E. E. 2020. All you need is low (rank) defending against adversarial attacks on graphs. In *WSDM*, 169–177.

Gasteiger, J.; Bojchevski, A.; and Günnemann, S. 2018. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *NeurIPS*, 30.

Katzir, Z.; and Elovici, Y. 2021. Gradients Cannot Be Tamed: Behind the Impossible Paradox of Blocking Targeted Adversarial Attacks. *IEEE TNNLS*, 32(1): 128–138.

Kenney, M. J.; and Ganta, C. K. 2014. Autonomic nervous system and immune system interactions. *Comprehensive Physiology*, 4(3): 1177.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Kipnis, J. 2016. Multifaceted interactions between adaptive immunity and the central nervous system. *Science*, 353(6301): 766–771.

Kreps, S.; and Kriner, D. 2020. Model uncertainty, political contestation, and public trust in science: Evidence from the COVID-19 pandemic. *Science Advances*, 6(43): eabd4563.

Li, K.; Liu, Y.; Ao, X.; Chi, J.; Feng, J.; Yang, H.; and He, Q. 2022. Reliable Representations Make A Stronger Defender: Unsupervised Structure Refinement for Robust GNN. *KDD*.

Li, Y.; Jin, W.; Xu, H.; and Tang, J. 2021. Deeprobust: A pytorch library for adversarial attacks and defenses. *AAAI*.

Liu, A.; Li, B.; Li, T.; Zhou, P.; and Wang, R. 2022. AN-GCN: An Anonymous Graph Convolutional Network Against Edge-Perturbing Attacks. *IEEE TNNLS*.

Liu, A.; Li, W.; Li, T.; Li, B.; Huang, H.; and Zhou, P. 2024. Towards Inductive Robustness: Distilling and Fostering Wave-Induced Resonance in Transductive GCNs against Graph Adversarial Attacks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 13855–13863.

Liu, J.; Akhtar, N.; and Mian, A. 2020. Adversarial Attack on Skeleton-Based Human Action Recognition. *IEEE TNNLS*, 33(4): 1609 – 1622.

Liu, X.; Jin, W.; Ma, Y.; Li, Y.; Liu, H.; Wang, Y.; Yan, M.; and Tang, J. 2021. Elastic graph neural networks. In *ICML*, 6837–6849. PMLR.

Meng, L.; Bai, Y.; Chen, Y.; Hu, Y.; Xu, W.; and Weng, H. 2023. Devil in Disguise: Breaching Graph Neural Networks Privacy through Infiltration. In *CCS*, 1153–1167.

Morton, N. W.; Schlichting, M. L.; and Preston, A. R. 2020. Representations of common event structure in medial temporal lobe and frontoparietal cortex support efficient inference. *PNAS*, 117(47): 29338–29345.

Samuel, F., G; John, B., D; Joichi, I.; Jonathan, Z., L; Andrew, B., L; and Isaac, K., S. 2019. Adversarial attacks on medical machine learning. *Science*, 363(6433): 1287–1289.

Sixiao, Z.; Hongxu, C.; Xiangguo, S.; Yicong, L.; and Xu, G. 2022. Unsupervised Graph Poisoning Attack via Contrastive Loss Back-propagation. In *WWW*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *ICLR*.

Walt, W.; Jack, C.; and Christof, T. 2019. Adversarial explanations for understanding image classification decisions and improved neural network robustness. *Nature Machine Intelligence*, 1: 508—516.

Wang, B.; and Gong, N. Z. 2019. Attacking graph-based classification via manipulating the graph structure. In *CCS*, 2023–2040.

Wang, S.; He, L.; Cao, B.; Lu, C.-T.; Yu, P. S.; and Ragin, A. B. 2017. Structural deep brain network mining. In *KDD*, 475–484.

Wu, F.; Long, Y.; Zhang, C.; and Li, B. 2022. Linkteller: Recovering private edges from graph neural networks via influence analysis. In *SP*, 2005–2024. IEEE.

Wu, H.; Wang, C.; Tyshetskiy, Y.; Docherty, A.; Lu, K.; and Zhu, L. 2019. Adversarial examples for graph data: deep insights into attack and defense. In *IJCAI*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2021. A comprehensive survey on graph neural networks. *IEEE TNNLS*, 32(1): 4–24.

Xi, Z.; Pang, R.; Ji, S.; and Wang, T. 2021. Graph backdoor. In *USENIX Security*, 1523–1540.

Zheng, Q.; Zou, X.; Dong, Y.; Cen, Y.; Yin, D.; Xu, J.; Yang, Y.; and Tang, J. 2021. Graph Robustness Benchmark: Benchmarking the Adversarial Robustness of Graph Machine Learning. In *NeurIPS*.

Zhu, D.; Zhang, Z.; Cui, P.; and Zhu, W. 2019. Robust graph convolutional networks against adversarial attacks. In *KDD*, 1399–1407.

Zügner, D.; Akbarnejad, A.; and Günnemann, S. 2018. Adversarial attacks on neural networks for graph data. In *KDD*, 2847–2856.

Zügner, D.; and Günnemann, S. 2018. Adversarial Attacks on Graph Neural Networks via Meta Learning. In *ICLR*.