

# ROBUSTNESS IN TEXT-ATTRIBUTED GRAPH LEARNING: INSIGHTS, TRADE-OFFS, AND NEW DEFENSES

**Runlin Lei<sup>1</sup> Lu Yi<sup>1</sup> Mingguo He<sup>2</sup> Pengyu Qiu<sup>3</sup>**  
**Zhewei Wei<sup>1\*</sup> Yongchao Liu<sup>3\*</sup> Chuntao Hong<sup>3</sup>**

<sup>1</sup>Renmin University of China <sup>2</sup>National University of Singapore <sup>3</sup>Ant Group

{runlin\_lei, yilu, zhewei}@ruc.edu.cn, mingguo@nus.edu.sg  
{pengyu.qpy, yongchao.ly, chuntao.hct}@antgroup.com

## ABSTRACT

While Graph Neural Networks (GNNs) and Large Language Models (LLMs) are powerful approaches for learning on Text-Attributed Graphs (TAGs), a comprehensive understanding of their robustness remains elusive. Current evaluations are fragmented, failing to systematically investigate the distinct effects of textual and structural perturbations across diverse models and attack scenarios. To address these limitations, we introduce a unified and comprehensive framework to evaluate robustness in TAG learning. Our framework evaluates classical GNNs, robust GNNs (RGNNs), and GraphLLMs across ten datasets from four domains, under diverse text-based, structure-based, and hybrid perturbations in both poisoning and evasion scenarios. Our extensive analysis reveals multiple findings, among which three are particularly noteworthy: 1) models have inherent robustness trade-offs between text and structure, 2) the performance of GNNs and RGNNs depends heavily on the text encoder and attack type, and 3) GraphLLMs are particularly vulnerable to training data corruption. To overcome the identified trade-offs, we introduce SFT-auto, a novel framework that delivers superior and balanced robustness against both textual and structural attacks within a single model. Our work establishes a foundation for future research on TAG security and offers practical solutions for robust TAG learning in adversarial environments. Our code is available at: <https://github.com/Leirunlin/TGRB>.

## 1 INTRODUCTION

Text-attributed graphs (TAGs), which integrate structural links with rich text features, are foundational to applications from social networks to citation graphs (Wu et al., 2023; Wang et al., 2025a). While Graph Neural Networks (GNNs) have long been the prevailing approach for learning on TAGs, Graph Large Language Models (GraphLLMs) are emerging as a compelling paradigm, leveraging their advanced reasoning capabilities directly on graph-structured text. However, the robustness of these models remains a critical challenge. For instance, in high-stakes domains such as social and financial networks, adversaries can manipulate both graph structures and textual content, significantly degrading model performance. For example, adversaries can deploy deceptive social bots with engineered biographies and network patterns to influence public opinion (Wang et al., 2023). Similarly, in recommendation systems, attackers may craft fake user profiles with misleading textual attributes to promote targeted items (Nguyen et al., 2024). This dual vulnerability makes it uniquely difficult to secure TAG learning.

Despite its importance, existing robustness analyses remain fragmented. Early analyses of GNNs and Robust GNNs (RGNNs) relied on naive embeddings, largely overlooking the rich semantic information in natural language (Zheng et al., 2021). Conversely, recent attempts start explorations of the robustness of GraphLLMs, yet lack comprehensive comparisons among model families and

---

\*Zhewei Wei and Yongchao Liu are the Corresponding authors.

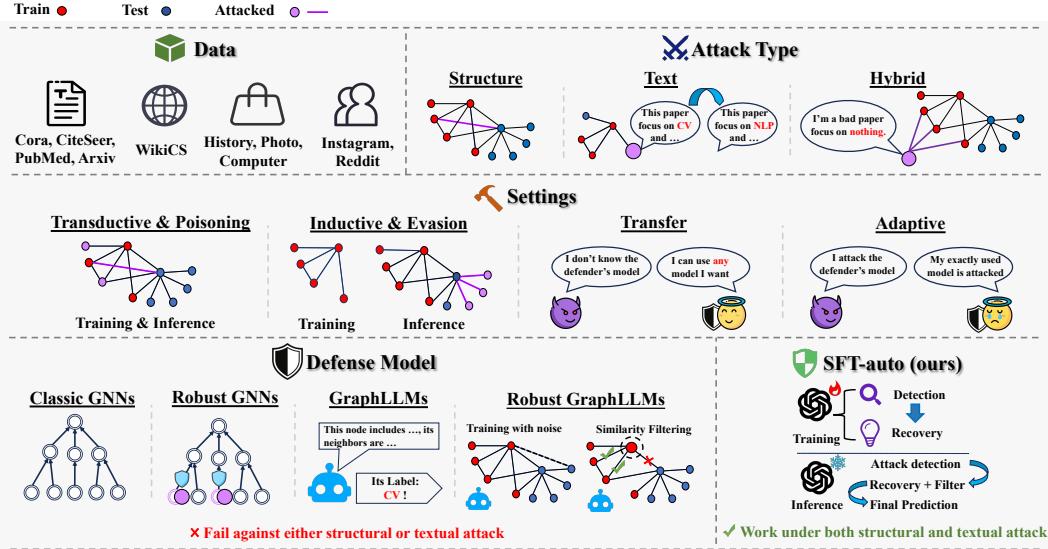


Figure 1: The overall framework for evaluating the robustness of TAG learning.

focus exclusively on limited attack settings (Guo et al., 2024a; Zhang et al., 2025a). This fragmented landscape has left the field without a comprehensive analysis of robust TAG learning.

To address this critical void, we perform a large-scale, systematic robustness analysis on TAGs. The overall framework is provided in Figure 1. Our evaluation spans ten datasets across four domains and a wide spectrum of models, including classical GNNs, RGNNs, and GraphLLMs. By subjecting these models to a unified threat model, we provide a unified evaluation methodology that serves as a foundation for understanding model vulnerabilities across diverse architectures and attack settings.

The extensive evaluation yields a series of empirical insights: 1) We uncover a crucial text-structure robustness trade-off, where models excel at defending against either textual or structural attacks but not both simultaneously. 2) We find that previously underrated methods, such as GNNGuard (Zhang & Zitnik, 2020), achieve surprising performance when re-evaluated in TAG settings with advanced text encoders. 3) GraphLLMs demonstrate higher vulnerability to poisoning attacks compared to GNNs. Specifically, when the training data is compromised, GraphLLMs experience a more significant decline in performance than GNNs. 4) Directly integrating existing robust GNN designs into LLM architectures fails to resolve the fundamental robustness trade-off.

Motivated by the proven effectiveness of noise-injection (Ennadir et al., 2024) and similarity-filtering strategies (Zhang & Zitnik, 2020; Hou et al., 2024) in RGNNs, we explore their adaptation to GraphLLMs to address the text-structure trade-off. While these variants show effectiveness against either textual or structural perturbations individually, they still struggle to achieve balanced robustness against both types of attacks. To achieve balanced robustness against both attacks, we propose a novel SFT (supervised fine-tuning) framework, SFT-auto, which employs multi-task training with a detection-prediction pipeline. This approach leverages the superior reasoning capabilities of LLMs to detect anomalies and make predictions within a single model. Our experiment results show that SFT-auto exhibits superior robustness in both modalities compared to the baselines.

To summarize, our **main contributions** are as follows:

- **A Comprehensive Evaluation Framework.** We propose a systematic robustness evaluation for learning in TAGs that benchmarks a wide spectrum of models, from classical GNNs, RGNNs, to GraphLLMs, against a diverse set of textual and structural attacks.
- **Abundant Empirical Insights.** Our large-scale analysis reveals critical vulnerabilities and trade-offs in robust TAG learning. We uncover a **text-structure robustness trade-off**, find that simple RGNNs with advanced text encoders can be surprisingly effective, and demonstrate the vulnerability of GraphLLMs to poisoning attacks.

- **An Effective Defense Framework.** To overcome the identified trade-off, we propose the SFT-auto model. It leverages the reasoning capabilities of LLMs to achieve superior and balanced robustness against both textual and structural attacks.

## 2 PRELIMINARIES: BACKGROUND AND EVALUATION PROTOCOL

### 2.1 BACKGROUND

We define a TAG as  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \{s_i\}_{i=1}^N)$ , where  $\mathcal{V}$  is the set of nodes,  $\mathcal{E}$  is the set of edges, and  $N$  is the number of nodes. The adjacency matrix is  $\mathbf{A} \in \{0, 1\}^{N \times N}$ , with each node  $v_i \in \mathcal{V}$  associated with a sequential text  $s_i$ . Following prior work on graph adversarial learning (Jin et al., 2021; Zheng et al., 2021), we focus on the task of node classification, where the goal is to assign each node one of  $C$  possible class labels. Denote the label vector as  $\mathbf{y} \in \{0, \dots, C-1\}^N$ , the learning objective is to predict labels of target nodes  $\mathbf{y}_{\text{target}}$ , given  $\mathcal{G}$  and ground-truth labels  $\mathbf{y}_{\text{train}}$ . In the **transductive setting**, all nodes are observed during training, while in the **inductive setting**, the model is required to generalize to previously unseen test nodes. Existing methods for TAG node classification include **GNNs** and **GraphLLMs**, denoted as a model  $f(\{s_i\}_{i=1}^N, \mathbf{A})$ . For GNNs, the model  $f$  first employs a text encoder to transform each  $s_i$  into a node-level feature matrix  $\mathbf{X}$ , and then processed jointly with the adjacency matrix  $\mathbf{A}$ . In contrast, some of the GraphLLMs directly input the raw node texts  $\{s_i\}_{i=1}^N$  into the model  $f$  and perform classification via prompt instructions.

**Graph Adversarial Attacks and Defenses.** In adversarial settings, an attacker seeks to degrade the performance of the defender’s model on a target set of nodes  $\mathcal{V}_{\text{target}}$ . A typical graph attack is the Graph Modification Attack (GMA) (Zügner & Günnemann, 2019; Xu et al., 2019), which perturbs either the graph structure or the node texts. The objective of GMA is:

$$\min_{\mathbf{A}', \{s'_i\}_{i=1}^N} \text{Acc}(f_{\theta}(\{s'_i\}_{i=1}^N, \mathbf{A}'), \mathbf{y}_{\text{target}}), \text{ s.t. } \|\mathbf{A}' - \mathbf{A}\|_0 \leq 2\Delta_{\text{struct}} \text{ and } \sum_{i=1}^N \mathbf{1}\{s'_i \neq s_i\} \leq \Delta_{\text{text}},$$

where  $\mathbf{A}'$  denotes the perturbed adjacency matrix,  $\{s'_i\}_{i=1}^N$  represents the perturbed node texts,  $\text{Acc}(\cdot)$  is the evaluation metric (e.g., accuracy) on the target nodes  $\mathcal{V}_{\text{target}}$ , and  $\Delta_{\text{struct}}$  and  $\Delta_{\text{text}}$  are the budget on the total number of perturbations. Specifically,  $\|\mathbf{A}' - \mathbf{A}\|_0$  measures the number of edge modifications (additions or deletions), and  $\sum_{i=1}^N \mathbf{1}\{s'_i \neq s_i\}$  counts the number of nodes with modified texts. Besides GMA, other paradigms targeting TAGs include Text-level Graph Injection Attacks (**Text-GIAs**), where new adversarial textual nodes are introduced into the graph, forming harmful connections to existing nodes (Lei et al., 2024).

Attacks can be categorized by their timing: **poisoning attacks** modify the training data to compromise the learned model, while **evasion attacks** alter test inputs to fool a fixed model at inference time. For the defender, the key objective is to maintain high performance even when the data may be under attack. Efforts have been made via RGNN design (Jin et al., 2021) and LLM as graph data purifiers (Zhang et al., 2025b).

Table 1: Comparisons between evaluations of robustness in TAG learning.

Benchmark	Data		Baselines			Evaluation Settings			
	Num. Datasets	Num. Domains	GNNs	RGNNs	GraphLLMs	Attack Types	Settings	Encoder Analysis	Adaptive Attack
GRB (Zheng et al., 2021)	5	2	✓	✓	✗	GIA	Evasion & Inductive	✗	✗
Gosch et al. (Gosch et al., 2023)	8	4	✓	✓	✗	GMA	Evasion & Both	✗	✓
Guo et al. (Guo et al., 2024a)	6	3	✓	✗	✓	GMA+Text	Evasion & Transductive	✓	✗
TrustGLM (Zhang et al., 2025a)	6	2	✗	✗	✓	GMA+Text+Prompt	Evasion & Transductive	✗	✗
Olatunji et al. (2025)	4	1	✗	✗	✓	GMA+Text	Both & Transductive	✗	✗
<b>Ours</b>	<b>10</b>	<b>4</b>	✓	✓	✓	GMA+Text+Text-GIA	All	✓	✓

### 2.2 EVALUATION PROTOCOL

Extensive evaluations of robustness in TAG learning have been developed. However, their analyses are limited, particularly with respect to data, baselines, and evaluation settings. As shown in Table 1, in terms of **data**, previous works suffer from limited dataset diversity and narrow domain

coverage, with some focusing exclusively on specific graph types or application domains. Regarding **baselines**, no previous work has achieved comprehensive integration of all three major graph learning paradigms (GNNs, RGNNS, and GraphLLMs), resulting in fragmented evaluation landscapes. Finally, **evaluation settings** in existing evaluations are often constrained by limited attack diversity.

To address these limitations, we include: (1) an extensive dataset diversity spanning multiple domains and graph types; (2) a unified comparison supporting major baseline categories (GNNs, RGNNS, GraphLLMs); and (3) extensive evaluation settings with comprehensive metrics. The detailed design of the evaluation follows the principles below:

- **Deploy sufficiently strong attacks.** We found that some attacks, such as Mettack (Zügner & Günnemann, 2019) and word-level textual attacks, don't generate sufficiently strong attacks or have poor transferability. Yet, weak perturbations fail to differentiate defense models, as performance rankings become dominated by clean accuracy rather than adversarial resilience (see Appendix D.4). Therefore, we employ more effective attacks with a sufficiently high perturbation ratio in the main paper to ensure a higher degree of differentiation. Results with smaller ratios are deferred to Appendix J.4.
- **Ensure fair baseline comparison.** We restrict evaluation to models with comparable clean performance to prevent stronger backbones from appearing artificially robust. Methods like Instruct-GLM, GPT zero-shot, and GraphPrompt in (Guo et al., 2024a; Zhang et al., 2025a; Olatunji et al., 2025) significantly underperform supervised baselines, invalidating robustness comparisons. Following established practices (Hou et al., 2024; Wu et al., 2025; Wang et al., 2025b), we select competitive GNNs, RGNNS, and GraphLLMs as baselines. The baselines are summarized in Table 2. The details of each baseline are provided in Appendix E.<sup>1</sup>
- **Adopt realistic evaluation protocols.** Prior benchmarks employ misaligned settings that compromise validity. As stated in (Gosch et al., 2023), poisoning attacks naturally pair with transductive learning, while evasion attacks suit inductive evaluation. Protocol misalignments enable trivial memorization-based defenses, undermining meaningful assessment. We strictly align attack types with appropriate learning paradigms across all experimental phases.

**Data.** We evaluate on ten datasets spanning four distinct domains following the LLMNodeBed benchmark (Wu et al., 2025): academic networks (Cora (Sen et al., 2008), CiteSeer (Giles et al., 1998), PubMed (Yang et al., 2016), ArXiv (Hu et al., 2020)), web links (WikiCS (Mernyei & Cangea, 2020)), social networks (Instagram, Reddit (Huang et al., 2024)), and e-commerce (History, Photo, Computer (Yan et al., 2023)). We adopt a supervised 60/20/20 split across training/validation/testing for the *inductive* setting and a semi-supervised 10/10/80 split for the *transductive* setting. All datasets are undirected graphs. Details of datasets are provided in Appendix C.

**Threat Model.** Our evaluation assesses perturbations to the graph structure, node texts, and also includes results against Text-GIAs (in Appendix F). We evaluate both poisoning attacks and evasion attacks. Our primary focus is on **transfer attacks** where the attacker has access to the victim's data but not their model directly. The perturbed graph is then transferred to test the defender's model, simulating a practical scenario where defenders can deploy their custom defense model. The specific attack configurations are detailed in the subsequent experimental sections, with comprehensive details provided in Appendix D. We also explore adaptive attacks in Appendix H.

**Other Setups.** We employ accuracy as the evaluation metric. Hyperparameters for all GNNs and RGNNS are optimized based on validation set performance. Following LLMNodeBed (Wu et al., 2025), we adopt ROBERTa (Liu et al., 2019) as the text encoder for GNN-based methods and Mistral-7B (Jiang et al., 2023) as the backbone for GraphLLMs, as these configurations yield optimal performance. All experiments are conducted across three independent runs with random data splits, except for ArXiv, which uses a single official split.

### 3 EVALUATION RESULTS

In this section, we present evaluation results against structural and textual attacks. Due to space limitations, we report the average rank, which is derived for each method by averaging its ranks across

<sup>1</sup>We move the evaluation of specific variants like GOOD-AT, GPR-GAE, and GPRGNN-AT to Appendix K. These supplementary results serve as an extended verification of our core findings.

Table 2: Categorization of selected defense models.

Taxonomy	Subcategory	Selected Defenses / Models
Basic models	Spatial / Message passing	GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2018)
	Spectral	APPNP (Klicpera et al., 2019), GPRGNN (Chien et al., 2021)
Improving training	Robust training	GRAND (Feng et al., 2020), NoisyGCN (Ennadir et al., 2024), Adversarial Training Gosch et al. (2023)
	Probabilistic	RobustGCN (Zhu et al., 2019)
	Similarity-based	GNNGuard (Zhang & Zitnik, 2020)
	Robust aggregation	ElasticGNN (Liu et al., 2021), SoftmedianGDC (Geisler et al., 2021), RUNG (Hou et al., 2024)
Improving architecture	Others	EvenNet (Lei et al., 2022) (Spectral), GCORN (Abbahaddou et al., 2024) (Weight Regularization),
	Unsupervised	Jaccard-GCN (Wu et al., 2019), Cosine-GCN (Mujkanovic et al., 2022)
	Supervised	ProGNN (Jin et al., 2020), Stable (Li et al., 2022b), GOOD-AT Li et al. (2024), GPR-GAE Lee & Park (2025)
GraphLLMs	Instruction Tuning & Align	GraphGPT (Tang et al., 2024), SFT (w/ nei.) (Wang et al., 2025b), LLaGA (Chen et al., 2024)

all datasets where it has valid results, with full numerical results available in Appendix J. We adopt rank-based evaluation because averaging raw accuracies across datasets can be misleading due to scale differences, missing results on large-scale graphs, and strong dependence on clean accuracy. Ranking provides a normalized, dataset-agnostic comparison that offers a clearer comparison. Similarly, the results about hybrid and adaptive attacks are deferred to Appendix F and H.

For structural attacks, in the inductive and evasion settings, we employ PGD (Xu et al., 2019) for small-scale datasets and GRBCD (Geisler et al., 2021) for larger ones with a perturbation ratio of 0.20. We use GCN (Kipf & Welling, 2017) as the surrogate model with BoW embeddings to generate victim graphs, and use the generated graphs as the test victim graphs. For *transductive* structural attacks, we adopt HeuristicAttack (Li et al., 2023) with a perturbation ratio of 0.30, and exclude Computer and ArXiv due to scalability issues.

We evaluate performance against textual attacks using a novel LLM-based attack. For evasion attacks, we substitute 40% of the test set nodes with LLM-generated text that differs from the original content. For poisoning attacks, we replace 80% of the training set nodes. Attack algorithm details are provided in Section D. Due to scalability concerns, we exclude the Computer and ArXiv datasets for evasion attacks, and additionally exclude the Photo dataset for poisoning attacks. We do not use gradient-based or unnoticeable word-level textual attacks, as these methods have been shown to have poor transferability across different models, as discussed in Appendix D.4.

### 3.1 AGAINST STRUCTURAL ATTACKS

The results against structural attacks are shown in Figure 2.

**GraphLLMs show Inherent Robustness against Non-adaptive Inductive/Evasion Attacks.** Even without defense mechanisms, SFT-neighbor outperforms most RGNNS. GraphGPT, which also employs instruction tuning, exhibits comparable strong performance, reflecting GraphLLM’s superior robustness against structural attacks. Notably, LLaGA shows relatively weaker robustness despite being a GraphLLM. Although LLaGA surpasses GraphGPT in clean perfor-

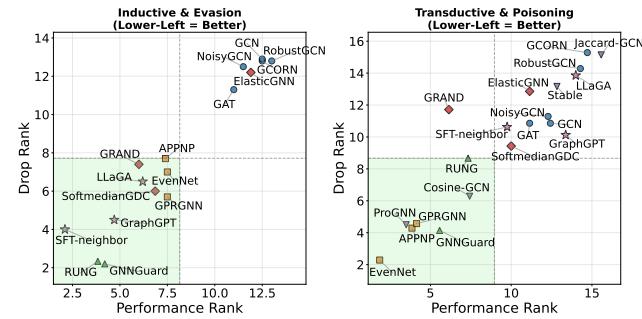


Figure 2: Comparison of robustness against structural attacks. Ranks are averaged across datasets (excluding failures) based on: 1) absolute accuracy under attack (lower rank is better), and 2) relative accuracy drop from the clean baseline (lower drop rank is better).

mance, it proves more vulnerable to structural attacks. This suggests that alignment-based methods, which explicitly utilize graph embeddings to align graph and text spaces, are more susceptible to structural attacks (similar to GCNs), while instruction-tuning models adopt more conservative neighbor utilization strategies.

**Simple Methods Can Shine in TAGs through Advanced Text Encoders.** Despite their simplicity, RGNNs from earlier research can deliver surprisingly strong performance when re-evaluated in TAGs. For instance, GNNGuard, as an early and straightforward RGNN that leverages threshold-based filtering to defend against adversarial attacks, achieves top-tier performance against the inductive/evasion attacks. This contrast stems from prior works that have overlooked the importance of text embeddings and have only evaluated RGNNs on shallow embeddings, such as BoW or TF-IDF (Lei et al., 2022; Hou et al., 2024). In the context of TAGs, methods like GNNGuard can be revitalized to achieve near-SOTA robustness. A detailed analysis is provided in Appendix G.

In fact, by fully harnessing textual features through dataset- and embedding-specific filtering, we can do even better. In Appendix G.3, we propose **Guardual**, a novel extension that eliminates the reliance of GNNGuard on threshold hyperparameters. The results show that Guardual’s adaptive filtering mechanism makes it the leading RGNN in the structural evasion setting. These findings underscore that strategic text processing in TAGs fundamentally drives RGNN performance.

**Spectral GNNs Show Superior Robustness against Poisoning Structural Attacks.** As shown in Figure 2 (right), spectral methods, such as EvenNet, APPNP, and GPRGNN, demonstrate superior performance against poisoning attacks in the transductive setting. This phenomenon complements the findings of Gosch et al. (2023) in the evasion/transductive setting, where the robust diffusion process in spectral methods enables flexible use of whole higher-order neighborhoods, thereby enhancing robustness. Structure learning methods, such as ProGNN, also exhibit promising results, though their computational overhead remains a significant practical limitation. In contrast, the performance of GraphLLMs starts to decline. While SFT-neighbor is robust against evasion attacks, the perturbations introduced during training result in a notable performance drop. Among GraphLLMs, LLaGA remains the most vulnerable due to its greater reliance on structure.

### 3.2 AGAINST TEXTUAL ATTACKS

The results against textual attacks are shown in Figure 3.

**Simple GNNs Excel Advanced Baselines.** When confronting textual attacks in the evasion setting, the relative robustness rankings undergo a notable shift. While methods like GNNGuard and RUNG demonstrate superior performance against structural attacks, as shown in Figure 2, they exhibit pronounced vulnerabilities to textual perturbations. SFT-neighbor and GraphGPT also suffer from significant performance degradation. In contrast, even naive models, such as GCN and GAT, exhibit the desired robustness against these textual attacks.

**GraphLLMs Struggle Against Textual Poisoning.** For poisoning attacks, both GNNs and RGNNs exhibit remarkable robustness. Even when 80% of the training nodes’ text is replaced, GNNs can still benefit from the transductive learning paradigm and achieve accurate predictions by aggregating information from nodes’ neighbors. Under text poisoning attacks, LLM-based methods experience a significant decline in performance. For example, on the CiteSeer dataset, SFT-neighbor’s accuracy drops by 25%, while most GNNs suffer only a 5%-10% decrease (see Tables 39 and 45 for detailed accuracy). This suggests that GraphLLMs are more vulnerable to perturbations in the training set, relying heavily on high-quality training text to maintain strong performance.



Figure 3: Comparison of robustness against textual attacks. Ranks are averaged across datasets (excluding failures) based on: 1) absolute accuracy under attack (lower rank is better), and 2) relative accuracy drop from the clean baseline (lower drop rank is better).

## 4 THE TEXT-STRUCTURE TRADE-OFF

In Section 3, we found that in the inductive/evasion setting, the structurally robust models fail against textual attacks (e.g., GNNGuard, SFT-neighbor); and vice versa (e.g., NoisyGCN). Building on this, we now explore how vulnerabilities in one dimension relate to the other.

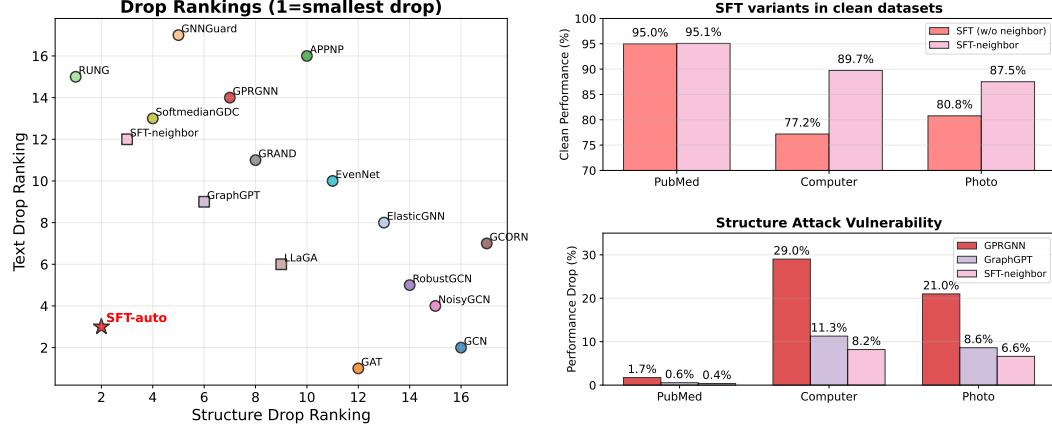


Figure 4: The text-structure robustness trade-off. **(Left)** SFT-auto uniquely balances this trade-off, while baseline models are polarized towards either text or structure robustness. **(Right)** On **text-friendly** datasets (e.g., PubMed), models are less reliant on structure and thus less vulnerable to its perturbation. The opposite holds for **structure-critical** datasets (e.g., Computer, Photo).

**Architecture Effect.** As shown in Figure 4 **(Left)**, models with different architectures exhibit a clear text-structure robustness trade-off. Structure-oriented architectures such as LLaGA and vanilla GNNs are highly vulnerable to structural perturbations yet remain comparatively stable under textual attacks. By contrast, text-oriented models like SFT-neighbor and GraphGPT, as well as RGNNs designed to enhance structural robustness, show strong resistance to structural attacks but collapse under textual ones. These comparisons highlight the inherent bias of different backbones: classifiers are generally robust to either structure or text perturbations, but rarely to both.

**Dataset Effect.** Dataset characteristics critically shape the text-structure trade-off. As shown in Figure 4 **(Right)**, the benefit of neighbor information differs markedly across datasets, directly impacting vulnerability. In the **top-right** panel, SFT variants gain substantial accuracy from neighbors on structure-critical datasets such as Computer/Photo, but little on text-friendly datasets like PubMed. This reliance pattern explains the robustness outcomes in the **bottom-right** panel: models relatively resistant to structural perturbations (e.g., GPRGNN, GraphGPT, SFT-neighbor) experience only minor drops on text-friendly datasets, yet still suffer significantly on structure-critical ones. Thus, while the trade-off is influenced by model design, its manifestation heavily depends on the nature of the datasets.

## 5 ADDRESSING THE TEXT-STRUCTURE ROBUSTNESS TRADE-OFF

The text-structure trade-off remains a key challenge, with no current model effectively balancing both aspects. In this section, we explore methods to overcome this limitation.

### 5.1 ATTEMPTS BY BUILDING ROBUST GRAPHLLMs

To address the text-structure trade-off, we explore noise-injection and similarity-filtering strategies inspired by existing RGNNs, adapting them for GraphLLMs.

**Noise-Injection Methods.** Drawing inspiration from GRAND (Feng et al., 2020) and NoisyGCN (Ennadir et al., 2024), we inject targeted perturbations during training to reduce the distribution gap between training and adversarial test conditions. We implement three variants: `-noise` (structural noise injection), `-noisetxt` (textual noise injection), and `-noisefull`

(combined noise types). All methods use a 10% noise ratio. GraphGPT-noisetxt and GraphGPT-noisefull are neglected due to poor clean performance.

Figure 5 demonstrates that targeted noise injection provides defense against corresponding attack types. The `-noise` variant helps improve structure robustness, while `-noisetxt` helps improve textual robustness. However, when the attack type is unknown, significant trade-offs emerge. Notably, `-noisefull` fails to achieve simultaneous defense against both attack types. None of the variants shows better results against both attacks. This limitation restricts noise-injection methods to specialized defenses for specific types of attacks.

**Similarity-Filtering Methods.** Inspired by the similarity-filtering strategy of GNNGuard, we propose two variants: `-simf`, which employs edge filtering similar to GNNGuard, and `-simp`, which modifies the SFT prompt to guide adaptive reliance—leveraging neighborhood information when text is unreliable and depending on textual content when structure is compromised. Figure 5 reveals that `-simf` exhibits behavior analogous to GNNGuard, providing effective defense against structural attacks but remaining vulnerable to text-based perturbations. In contrast, `-simp` yields modest robustness improvements without significant performance gains. This suggests that simple instruction modifications are insufficient for LLMs to learn effective defenses that break the identified trade-off and learn effective defenses.

## 5.2 AUTO FRAMEWORK FOR ADVERSARIAL ATTACK DETECTION AND RECOVERY

The preliminary attempts highlight the fundamental challenge of achieving balanced robustness against both types of attacks. As a solution, we propose `SFT-auto`, a novel model that leverages the reasoning abilities of LLMs to defend against both textual and structural adversarial attacks.

**Training.** The training phase employs a principled data augmentation strategy to endow the model with attack recognition and recovery capabilities. We use an adaptive attack ratio  $r = \min(1/|\mathcal{C}|, 0.15)$  to ensure balanced detection across datasets with varying class distributions. Training data comprises three distinct types: **Normal samples** ( $\mathcal{S}_{\text{normal}}$ ) preserve original node-neighbor pairs to maintain standard classification ability; **Attack samples** ( $\mathcal{S}_{\text{attack}}$ ) contain nodes with text deliberately replaced by content from different-class nodes, labeled as “text\_attacked” to teach attack recognition; and **Recovery samples** ( $\mathcal{S}_{\text{recovery}}$ ) remove center text entirely, compelling the model to leverage neighbor information for robust prediction. This training paradigm enables the LLM to handle  $(|\mathcal{C}| + 1)$ -class attack detection and  $|\mathcal{C}|$ -class recovery tasks through specialized prompts.

**Inference.** The inference phase implements a three-stage adaptive pipeline that dynamically responds to detected attack patterns. **Stage 1: Attack Detection:** The LLM identifies text-attacked nodes through an extended  $(|\mathcal{C}| + 1)$ -dimensional classification space, while structure attacks are detected via embedding-based similarity analysis. Nodes exhibiting low cosine similarity ( $< 0.5$ ) to over half their neighbors are flagged as structure-attacked. Text attack detection takes precedence to prevent redundant dual flagging. **Stage 2: Adaptive Recovery:** Text-attacked nodes bypass their corrupted center text entirely, relying solely on original neighbor information for classification. Structure-attacked nodes leverage their preserved own text combined with filtered neighbors. Connections to text-attacked nodes or those with low similarity are removed. Normal nodes employ standard classification using their original text and neighbors, with only text-attacked neighbors filtered. The detailed pseudo-code of `SFT-auto` is given in Algorithm 1.

**Complexity Analysis.** The computational cost of `SFT-auto` is comparable to `SFT-neighbor`, as both methods are bottlenecked by the per-sample forward pass through the LLM. Training requires at most  $1.3 \times$  more samples due to data augmentation (with ratio  $r \leq 0.15$ ). For inference, let  $\mathcal{T}_{\text{LLM}}$  be the time for a single forward pass. The average per-sample inference time for `SFT-auto` is  $\mathcal{T}_{\text{avg}} \approx$

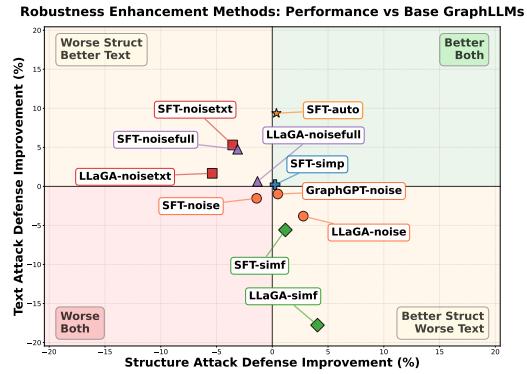


Figure 5: Performance of robust variants against base GraphLLMs. Points represent improvements over base models against structural (x-axis) and textual (y-axis) evasion attacks.

$(1 + p_{\text{attack}}) \cdot \mathcal{T}_{\text{LLM}}$ , where  $p_{\text{attack}}$  is the small fraction of detected nodes requiring recovery. The value of  $p_{\text{attack}}$  is bounded above by 2, which incurs an acceptable worst-case overhead, and is typically very small in practice. Given the equivalent per-sample cost, SFT-auto’s overall runtime is comparable to its baseline, making it an efficient framework for achieving balanced robustness.

**Results.** SFT-auto demonstrates superior performance, as visualized in Figure 4. Compared to baselines, SFT-auto has more consistent performance against both structural and textual attacks.

### 5.3 GNN WITH AUTO DESIGN

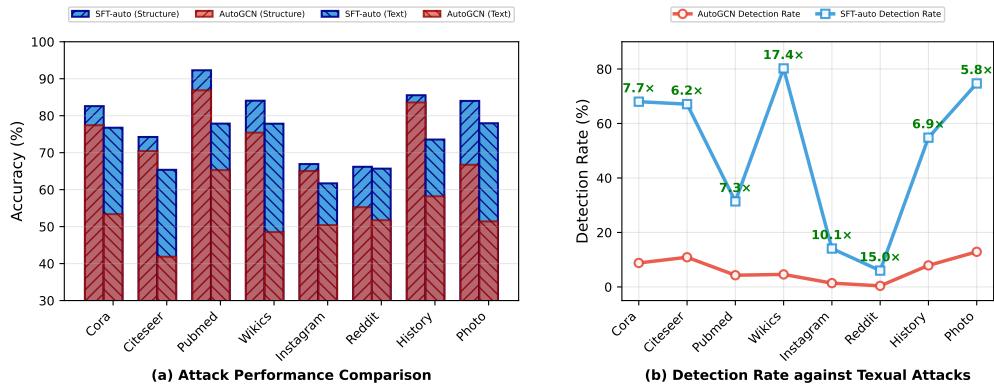


Figure 6: Performance comparison between SFT-auto and AutoGCN. **(a)** Accuracy under structural (left bars) and textual (right bars) attacks. **(b)** Detection efficacy against textual attacks.

Having established SFT-Auto’s effectiveness, a natural question arises: *Can similar auto-detection principles enhance the robustness of GNNs?* To explore this, we implement AutoGCN by replacing the LLM predictor with a GCN, while maintaining the same detection pipeline architecture.

Figure 6 reveals that AutoGCN exhibits substantial degradation across both attack modalities, with particularly pronounced deficiencies in textual anomaly detection (Figure 6b), where SFT-Auto achieves 6.2–17.4× improvements over AutoGCN. This performance gap illuminates fundamental architectural differences: LLMs possess inherent multi-modal reasoning capabilities, enabling seamless integration of detection and classification within unified frameworks. Conversely, GNNs lack the linguistic sophistication required for text anomaly detection. This suggests promising research directions toward hybrid architectures that combine GNNs’ structural robustness with LLMs’ semantic understanding through multi-stage prediction pipelines built on verified, clean data.

## 6 RELATED WORK

**Robust GNNs and GraphLLMs.** The vulnerability of GNNs to adversarial attacks has been extensively studied (Jin et al., 2021; Xue et al., 2025), motivating a series of RGNNs. Robust training methods modify the learning process to enhance resilience, exemplified by GRAND (Feng et al., 2020) and NoisyGCN (Ennadir et al., 2024). Robust architectural designs introduce inherently stable mechanisms, such as GNNGuard (Zhang & Zitnik, 2020), which filters edges based on node similarity, SoftMedian-GDC (Geisler et al., 2021), which applies median-based aggregation, and RUNG (Hou et al., 2024), which adopts an unbiased aggregator for improved soft filtering. Graph structure learning methods, including ProGNN (Jin et al., 2020) and Stable (Li et al., 2022b), refine the graph topology to denoise adversarial input. However, these RGNNs rely solely on shallow embeddings, neglecting the influence of raw textual information in TAGs on robustness, and focus primarily on defending against structural perturbations.

Recently, LLMs have been introduced for robust TAG learning. Representative methods include GraphEdit (Guo et al., 2024b), RLLMGNN (Zhang et al., 2025b), and LangGSL (Su et al., 2024), which leverage LLMs to adjust or reconstruct graph structure under adversarial settings. However, these approaches use LLMs solely as structure refiners and remain tightly coupled to GNN backbones, limiting their capacity to capture deeper interactions between text and structure.

**Graph Robustness Evaluation.** For GNNs and RGNNs, Zheng et al. (2021) propose GRB, which evaluates model robustness under GIAs. More recently, Guo et al. (2024a) initiate the study of robustness for LLM-based predictors in TAGs. TrustGLM (Zhang et al., 2025a) extends GraphLLM as the evaluation target and explores defense strategies such as noise injection. Olatunji et al. (2025) propose a deep evaluation into GraphLLMs against text and structural attacks. However, existing evaluation frameworks lack uniformity and fairness across different model categories and attack settings. This limitation obscures the key findings presented in our study.

## 7 CONCLUSION

This paper presents a comprehensive evaluation of graph learning methods on TAGs against both textual and structural attacks, evaluating GNNs, RGNNs, GraphLLMs, and LLMs across ten datasets from four domains under transductive poisoning and inductive evasion settings. The experiments reveal key insights: different classifier types exhibit distinct text-structure trade-offs; simple RGNN can shine again with a proper text encoder; and GNNs and LLMs demonstrate vulnerabilities to different attack types. The paper presents a novel method, SFT-auto, to address the identified trade-off, introducing a unified LLM-based framework that is robust against both textual and structural perturbations. The paper also includes comprehensive ablation studies and evaluations against adaptive and hybrid attacks in the appendix, establishing a foundation for future research in TAG security.

## REPRODUCIBILITY STATEMENT

We provide implementation details, configurations, and model cards in Appendix B. We provide detailed dataset descriptions in Appendix C. We provide complete specifications and hyperparameters of the attacks in Appendix D, and descriptions of all defense models, hyperparameters, and variants in Appendix E.

We report the full numerical results underlying the main text, including per-dataset, per-model, and per-attack breakdowns in Appendix J, enabling exact comparison and re-analysis.

## ACKNOWLEDGEMENTS

The work was partially done at Gaoling School of Artificial Intelligence, Beijing Key Laboratory of Research on Large Models and Intelligent Governance, Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE, and Pazhou Laboratory (Huangpu), Guangzhou, Guangdong 510555, China. This research was supported in part by National Natural Science Foundation of China (No. 92470128, No. U2241212), by National Science and Technology Major Project (2022ZD0114802), by Beijing Outstanding Young Scientist Program No.BJJWZYJH012019100020098, National Key Research and Development Plan of China (2023YFB4502305), and Ant Group through CCF-Ant Research Fund. We also wish to acknowledge the support provided by the fund for building world-class universities (disciplines) of Renmin University of China, by Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education, by Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the “Double-First Class” Initiative, Public Policy and Decision-making Research Lab, and Public Computing Cloud, Renmin University of China.

## REFERENCES

Yassine Abbahaddou, Sofiane Ennadir, Johannes F. Lutzeyer, Michalis Vazirgiannis, and Henrik Boström. Bounding the expected robustness of graph neural networks subject to node feature attacks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. Lлага: Large language and graph assistant. *arXiv preprint arXiv:2402.08170*, 2024.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

Sofiane Ennadir, Yassine Abbahaddou, Johannes F Lutzeyer, Michalis Vazirgiannis, and Henrik Boström. A simple and yet fairly effective defense for graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 21063–21071, 2024.

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103, 2020.

Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. *Advances in Neural Information Processing Systems*, 34:7637–7649, 2021.

C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: an automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pp. 89–98, 1998.

Lukas Gosch, Simon Geisler, Daniel Sturm, Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Adversarial training for graph neural networks: Pitfalls, solutions, and new directions. *Advances in neural information processing systems*, 36:58088–58112, 2023.

Kai Guo, Zewen Liu, Zhikai Chen, Hongzhi Wen, Wei Jin, Jiliang Tang, and Yi Chang. Learning on graphs with large language models(lms): A deep dive into model robustness. *CoRR*, abs/2407.12068, 2024a. doi: 10.48550/ARXIV.2407.12068. URL <https://doi.org/10.48550/arXiv.2407.12068>.

Zirui Guo, Lianghao Xia, Yanhua Yu, Yuling Wang, Kangkang Lu, Zhiyong Huang, and Chao Huang. Graphedit: Large language models for graph structure learning. *arXiv preprint arXiv:2402.15183*, 2024b.

Zhichao Hou, Ruiqi Feng, Tyler Derr, and Xiaorui Liu. Robust graph neural networks via unbiased aggregation. *Advances in Neural Information Processing Systems*, 37:110097–110130, 2024.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanming Tao, Zixing Chai, and Qi Zhu. Can gnn be good adapter for llms? *Proceedings of the ACM Web Conference 2024*, pp. 2077–2088, 2024.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. *Mistral 7b*, 2023. URL <https://arxiv.org/abs/2310.06825>.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 66–74, 2020.

Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, 22(2):19–34, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL <https://openreview.net/forum?id=SUJU4ayYgl>.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL <https://openreview.net/forum?id=H1gL-2A9Ym>.

Woohyun Lee and Hogun Park. Self-supervised adversarial purification for graph neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Runlin Lei, Zhen Wang, Yaliang Li, Bolin Ding, and Zhewei Wei. Evennet: Ignoring odd-hop neighbors improves robustness of graph neural networks. *Advances in neural information processing systems*, 35:4694–4706, 2022.

Runlin Lei, Yuwei Hu, Yuchen Ren, and Zhewei Wei. Intruding with words: Towards understanding graph injection attacks at the text level. *Advances in Neural Information Processing Systems*, 37: 49214–49251, 2024.

Jintang Li, Bingzhe Wu, Chengbin Hou, Guoji Fu, Yatao Bian, Liang Chen, Junzhou Huang, and Zibin Zheng. Recent advances in reliable deep graph learning: Inherent noise, distribution shift, and adversarial attack. *arXiv preprint arXiv:2202.07114*, 2022a.

Kuan Li, Yang Liu, Xiang Ao, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. Reliable representations make a stronger defender: Unsupervised structure refinement for robust gnn. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 925–935, 2022b.

Kuan Li, Yang Liu, Xiang Ao, and Qing He. Revisiting graph adversarial attack and defense from a data distribution perspective. In *The Eleventh International Conference on Learning Representations ICLR*, 2023.

Kuan Li, YiWen Chen, Yang Liu, Jin Wang, Qing He, Minhao Cheng, and Xiang Ao. Boosting the adversarial robustness of graph neural networks: An ood perspective. In *The Twelfth International Conference on Learning Representations*, 2024.

Xiaorui Liu, Wei Jin, Yao Ma, Yaxin Li, Hua Liu, Yiqi Wang, Ming Yan, and Jiliang Tang. Elastic graph neural networks. In *International Conference on Machine Learning*, pp. 6837–6849. PMLR, 2021.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

Mistral AI Team. Un ministral, des ministraux: Introducing ministral 3b and ministral 8b. <https://mistral.ai/news/ministraux>, October 2024.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 119–126, 2020.

Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. Are defenses for graph neural networks robust? *Advances in Neural Information Processing Systems*, 35: 8954–8968, 2022.

Thanh Toan Nguyen, Nguyen Quoc Viet Hung, Thanh Tam Nguyen, Thanh Trung Huynh, Thanh Thi Nguyen, Matthias Weidlich, and Hongzhi Yin. Manipulating recommender systems: A survey of poisoning attacks and countermeasures. *ACM Computing Surveys*, 57(1):1–39, 2024.

Iyiola E Olatunji, Franziska Boenisch, Jing Xu, and Adam Dziedzic. Adversarial attacks and defenses on graph-aware large language models (llms). *arXiv preprint arXiv:2508.04894*, 2025.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–106, 2008.

Guangxin Su, Yifan Zhu, Wenjie Zhang, Hanchen Wang, and Ying Zhang. Bridging large language models and graph structure learning models for robust representation learning. *arXiv preprint arXiv:2410.12096*, 2024.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 491–500, 2024.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL <https://openreview.net/forum?id=rJXMpikCZ>.

Lanjun Wang, Xinran Qiao, Yanwei Xie, Weizhi Nie, Yongdong Zhang, and Anan Liu. My brother helps me: Node injection based adversarial attack on social bot detection. In *Proceedings of the 31st ACM International Conference on Multimedia*, pp. 6705–6714, 2023.

Shijie Wang, Jian Huang, Zhikai Chen, Yu Song, Wenzhuo Tang, Haitao Mao, Wenqi Fan, Hui Liu, Xiaorui Liu, Dawei Yin, et al. Graph machine learning in the era of large language models (llms). *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–40, 2025a.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788, 2020.

Yuxiang Wang, Xinnan Dai, Wenqi Fan, and Yao Ma. Exploring graph tasks with pure llms: A comprehensive benchmark and investigation. *arXiv preprint arXiv:2502.18771*, 2025b.

Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. In Sarit Kraus (ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 4816–4823. ijcai.org, 2019.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, 55(5):97:1–97:37, 2023.

Xixi Wu, Yifei Shen, Fangzhou Ge, Caihua Shan, Yizhu Jiao, Xiangguo Sun, and Hong Cheng. When do llms help with node classification? a comprehensive analysis. *CoRR*, abs/2502.18771, 2025. doi: 10.48550/ARXIV.2502.18771. URL <https://doi.org/10.48550/arXiv.2502.18771>.

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. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 3961–3967. ijcai.org, 2019.

Ruizhan Xue, Huimin Deng, Fang He, Maojun Wang, and Zeyu Zhang. Trustworthy gnns with llms: A systematic review and taxonomy. *arXiv preprint arXiv:2502.08353*, 2025.

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, et al. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 36:17238–17264, 2023.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48, 2016.

Qihai Zhang, Xinyue Sheng, Yuanfu Sun, and Qiaoyu Tan. Trustglm: Evaluating the robustness of graphllms against prompt, text, and structure attacks. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 5912–5923, 2025a.

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

Zhongjian Zhang, Xiao Wang, Huichi Zhou, Yue Yu, Mengmei Zhang, Cheng Yang, and Chuan Shi. Can large language models improve the adversarial robustness of graph neural networks? In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 2008–2019, 2025b.

Qinkai Zheng, Xu Zou, Yuxiao Dong, Yukuo Cen, Da Yin, Jiarong Xu, Yang Yang, and Jie Tang. Graph robustness benchmark: Benchmarking the adversarial robustness of graph machine learning. *arXiv preprint arXiv:2111.04314*, 2021.

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*, pp. 1399–1407, 2019.

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. URL <https://openreview.net/forum?id=Bylnx209YX>.

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*, pp. 2847–2856, 2018.

# Contents

<b>A Usage of LLMs</b>	<b>17</b>
<b>B Implementation and Configurations</b>	<b>17</b>
B.1 Implementation . . . . .	17
B.2 Configuration . . . . .	17
B.3 Model Cards . . . . .	17
<b>C Details of Datasets</b>	<b>17</b>
<b>D Details of Attack Methods</b>	<b>18</b>
D.1 Structural Attacks . . . . .	18
D.2 Text-based Attacks . . . . .	19
D.3 Hybrid Text-level Graph Injection Attacks . . . . .	20
D.4 Attacks Excluded in the Main Paper . . . . .	20
D.4.1 Why not Mettack? . . . . .	20
D.4.2 Why not TextAttack? . . . . .	21
<b>E Details of Defense Methods</b>	<b>23</b>
E.1 Introduction of Defense Model . . . . .	23
E.2 Configuration and Hyperparameters . . . . .	24
E.2.1 GNN and RGNNs . . . . .	24
E.2.2 GraphLLMs . . . . .	25
E.2.3 GraphLLMs with Noisy Training . . . . .	26
E.2.4 SFT with Similarity Constraints . . . . .	27
E.2.5 SFT-auto . . . . .	27
<b>F Results against WTGIA</b>	<b>28</b>
<b>G Why GNNGuard Excels: Understanding and Improving</b>	<b>30</b>
G.1 Results of GNNGuard under different text encoders . . . . .	30
G.2 Difference Between Text-encoders . . . . .	30

---

G.3	Guardual: Robustness Against Test-Time Structural Attacks	32
G.3.1	Dual-Threshold Design	33
G.3.2	Results of Guardual	33
<b>H</b>	<b>Adaptive Attacks</b>	<b>33</b>
H.1	Adaptive Attacks against GNNGuard	33
H.2	Adaptive Attacks on Embedding Transferability	35
<b>I</b>	<b>Ablation Study for SFT Variants</b>	<b>36</b>
I.1	Impact of Different Prompt Templates	36
I.1.1	Degree-based vs Random Neighbor Selection	36
I.1.2	Label Information in Prompting	36
I.2	Impact of Different LLM Backbones	36
<b>J</b>	<b>Full Experiment Results</b>	<b>38</b>
J.1	Overview and Clarification of Rank-based Evaluation	38
J.2	GNN and RGNN Results	38
J.2.1	Clean, Inductive	38
J.2.2	Clean, Transductive	39
J.2.3	Structure Attack, Inductive	39
J.2.4	Structure Attack, Transductive	39
J.2.5	Text Attack, Inductive	39
J.2.6	Text Attack, Transductive	39
J.3	GraphLLM Results	44
J.4	Results with Different Perturb Ratios	44
<b>K</b>	<b>Additional Baseline Evaluation</b>	<b>48</b>

## A USAGE OF LLMs

In this study, Large Language Models (LLMs) were employed exclusively for the purpose of linguistic enhancement, including grammatical corrections and stylistic improvements. The conception, development, and interpretation of all technical material, experimental designs, analytical processes, and findings were conducted without any contribution from LLMs. The authors affirm their complete accountability for the entire content of this work.

## B IMPLEMENTATION AND CONFIGURATIONS

### B.1 IMPLEMENTATION

We rely on GreatX Li et al. (2022a) for GNN and RGNN implementations, and on NodeBed Wu et al. (2025) for GraphLLM and dataset loading.

### B.2 CONFIGURATION

Experiments were run on a machine with an NVIDIA A100-SXM4 GPU (80 GB), an Intel Xeon CPU (2.30 GHz), and 512 GB of RAM.

### B.3 MODEL CARDS

We used the following public models (links to their official model cards):

- Mistral-7B (Jiang et al., 2023)
- Llama 3.1–8B (Dubey et al., 2024)
- Minstral-8B (Mistral AI Team, 2024)
- Qwen3–8B (Yang et al., 2025)
- GPT-4o mini (Hurst et al., 2024)
- RoBERTa (Liu et al., 2019)
- MiniLM (Wang et al., 2020)

## C DETAILS OF DATASETS

We evaluate our methods on 10 text-attributed graph datasets spanning four domains, selected from the LLMNodeBed benchmark (Wu et al., 2025) to ensure comprehensive coverage of real-world scenarios.

### Academic Networks

- **Cora** (Sen et al., 2008): Computer science papers organized into seven research areas.
- **CiteSeer** (Giles et al., 1998): CS publications spanning six categories including AI and databases.
- **Pubmed** (Yang et al., 2016): Biomedical literature focused on diabetes research with three classification types.
- **ArXiv** (Hu et al., 2020): Large-scale CS paper collection covering 40 specialized subcategories from the arXiv repository.

### Web Link Network

- **WikiCS** (Mernyei & Cangea, 2020): Computer science Wikipedia articles categorized into ten technical domains, interconnected through hyperlink references.

### Social Networks

- **Instagram** (Huang et al., 2024): User profiles differentiated between personal and business accounts based on profile characteristics.
- **Reddit** (Huang et al., 2024): Community users classified by engagement levels using posting history and interaction patterns.

### E-Commerce Networks

- **History** (Yan et al., 2023): Historical literature products with detailed categorical organization.
- **Photo** (Yan et al., 2023): Photography equipment spanning professional and consumer categories.
- **Computer** (Yan et al., 2023): Technology products including hardware components and accessories.

Table 3: Dataset statistics for the 10 evaluated datasets from LLMNodeBed (Wu et al., 2025).

Domain	Dataset	Classes	Nodes	Edges
Academic	Cora	7	2,708	5,429
	CiteSeer	6	3,186	4,277
	Pubmed	3	19,717	44,338
	arXiv	40	169,343	1,166,243
Web Link	WikiCS	10	11,701	215,863
Social	Instagram	2	11,339	144,010
	Reddit	2	33,434	198,448
E-Commerce	History	12	41,551	358,574
	Photo	12	48,362	500,928
	Computer	10	87,229	721,081

## D DETAILS OF ATTACK METHODS

### D.1 STRUCTURAL ATTACKS

**PGD Attack (Xu et al., 2019).** A white box gradient-based discrete GMA that iteratively perturbs the graph structure via projected gradient ascent over continuous relaxation variables, followed by stochastic binarization to apply edge additions/removals under a budget constraint.

*Hyperparameters:*

- Learning rate:  $\eta_0 = 0.1$ .
- Optimization: 200 optimization epochs + 20 sampling epochs.
- Budget: Default 20% of total edges.
- Target Embedding: BoW.

**PGD-Guard (Threshold-based PGD) (Mujkanovic et al., 2022).** An adaptive variant that constrains perturbations to pairs of nodes whose cosine similarity exceeds a threshold, emulating defense-aware strategies intended to bypass similarity filtering (targeting GNNGuard (Zhang & Zitnik, 2020)).

*Hyperparameters:*

- Cosine similarity thresholds: [0.0, 0.3, 0.5, 0.7].
- Base settings: same as standard PGD.

- Budget: 20% of edges.
- Target Embedding: RoBERTa.

**GRBCD Attack (Geisler et al., 2021).** A white-box GMA based on greedy randomized block coordinate descent over the discrete edge space. At each step, random edge blocks are scored by gradients and greedily flipped within the budget.

*Hyperparameters:*

- Block size: 1,000,000.
- Sampling: 50 trials per iteration; 20 final samples for selection.
- Early stopping: patience-based with tolerance  $\epsilon = 10^{-7}$ .
- Target Embedding: BoW.

**HeuristicAttack (Li et al., 2023).** A scalable DICE-style heuristic (“Disconnect Internally, Connect Externally”) with training-aware constraints that prioritizes edges involving training nodes and degree-based node selection, approaching gray-box poisoning MetaAttack (Zügner & Günnemann, 2019) performance via distribution shifts maximization.

*Hyperparameters:*

- Add vs. remove probability: 0.5. We tried from [0.3, 0.5, 0.7, 0.9, 1.0], and found threshold= 0.5 yields most stable performance.
- Node sampling: inverse-degree probability (lower degree  $\Rightarrow$  higher probability).

## D.2 TEXT-BASED ATTACKS

**LLM Text Attack.** We generate neighborhood-aware prompts to induce an LLM to rewrite node texts so that the predicted label is driven away from (i) the node’s current class and (ii) the dominant classes in its immediate neighborhood, while preserving length and fluency.

*Prompt Template (instantiated per target node).*

**Graph node classification task**  
 Available classes: {classes\_str}  
 Target node {node\_id};  
 Original text: "{text}"  
 Original label: "{current\_label}"  
 {neighbor\_info} (e.g., “Neighbor labels: [...] (counts: [...])” or “No neighbors found”)

**Task.** Rewrite the text to be as different as possible from the original while keeping a similar length.

**Requirements:**

- Must *not* belong to the original class: “{current\_label}”.
- Should *not* belong to neighbor classes: {unique\_neighbor\_labels} or None.
- {target\_instruction} (e.g., *prefer a class from the allowed set, or the least frequent neighbor class if all are forbidden*).
- Make the content maximally dissimilar from the original semantics.
- Keep the word count roughly similar.
- Produce content that is most unlikely under the node/neighbor context for the target class.

Goal: create text that is jointly inconsistent with the original content and its local graph context.

Return *only* the modified text, with no explanations or notes.

**Modified text:**

*Algorithm.*

- **Target selection:** sample nodes with degree-weighted probabilities. Lower degrees, higher probabilities.

- **Context extraction:** for each node, gather its original text, current label, neighbors’ labels, and counts.
- **Constraint synthesis:** define the forbidden set as  $\{\text{current label}\} \cup \{\text{neighbor labels}\}$ ; compute the allowed label set over all classes. If empty, choose the least frequent neighbor label; else prefer a label maximally different from the forbidden set.
- **Prompting:** instantiate the above template with  $\{\text{classes\_str}\}$ ,  $\{\text{node\_id}\}$ ,  $\{\text{text}\}$ ,  $\{\text{current\_label}\}$ ,  $\{\text{neighbor\_info}\}$ , and  $\{\text{target\_instruction}\}$ .
- **Generation:** query the LLM to generate a response.
- **Post-processing:** clean text, validate constraints (length and class-avoidance heuristics), and embed to update node features.

*Hyperparameters.*

- Backbone LLM: GPT-4o-mini (Hurst et al., 2024).
- Temperature: 0.7.
- Target nodes: Training if under the poison setting, test nodes if under the evasion setting.
- Budget: Poison: 80% of the training nodes; Evasion: 40% of the test nodes.

### D.3 HYBRID TEXT-LEVEL GRAPH INJECTION ATTACKS

**WTGIA: Word-level Text GIA (Lei et al., 2024).** We follow the original WTGIA pipeline with dataset-specific adaptations during inductive learning.

*Configuration.*

- Text generator: Llama-3.1-8B (Dubey et al., 2024) with no-topic prompts and vocabulary masking.
- Edge connectivity: We use  $n_{\text{inject\_edges}} = \text{num\_edges} \times \text{ptb\_rate}$ , so the number of total inject edges aligned to GMA rates if the budget is set the same. Specifically, we yield 17 (Cora), 9 (CiteSeer), 22 (PubMed) at  $\text{ptb\_rate} = 0.2$ .
- Node injections: 60 (Cora), 90 (CiteSeer), 400 (PubMed) at  $\text{ptb\_rate} = 0.2$ ; scale proportionally for  $\text{ptb\_rate} = 0.4$ .
- BoW sparsity: 0.15 (Best according to the original paper).
- FGSM optimization: step size  $\epsilon = 0.01$  for 100 epochs; sequential injection steps of 0.2 with ATDGIA strategy.
- Batching: 50 for PubMed; 1 for other datasets.

### D.4 ATTACKS EXCLUDED IN THE MAIN PAPER

In this paper, we focus on untargeted attacks and text-based attacks. Therefore, methods such as Nettack (Zügner et al., 2018) and feature attacks in GRB (Zheng et al., 2021) are not employed. Additionally, while some attacks conform to our experimental setting, we choose not to adopt them. In this subsection, we provide detailed justification for these exclusions.

#### D.4.1 WHY NOT METTACK?

**Mettack (Zügner & Günnemann, 2019).** Mettack (Zügner & Günnemann, 2019) is a gray-box structural poisoning attack that employs a surrogate GCN and bi-level optimization with meta-gradients to identify vulnerable edges.

*Hyperparameters.*

- Surrogate learning rate: 0.1; momentum: 0.9.
- Meta learning rate: adaptive.
- Meta epochs: 100.

- Regularization modes:  $\lambda \in \{0.0 \text{ (meta-self)}, 0.5 \text{ (meta-both)}, 1.0 \text{ (meta-train)}\}$ . We use  $\lambda \in \{0.0 \text{ (meta-self)}\}$  as it yields the strongest attacks.

In the main paper, we opted for HeuristicAttack instead of Mettack for the following reasons:

- The attack performance of Mettack is not significant when transferring to validation-based defenses, as demonstrated in Table 4. This phenomenon is also evidenced by the GreatX repository (Li et al., 2022a).
- Mettack suffers from scalability limitations, making it applicable only to datasets of a size comparable to the PubMed level.

Table 4: Attack performance comparison on Cora and CiteSeer datasets using a surrogate GCN with 64 hidden units. Results show mean accuracy  $\pm$  standard deviation across three runs.

Dataset	Attack Method	Clean	No Validation	With Validation
Cora	HeuristicAttack	$82.91 \pm 0.83$	$61.30 \pm 1.00$	$70.33 \pm 2.89$
	Mettack	$82.91 \pm 0.83$	$75.13 \pm 1.28$	$77.86 \pm 0.08$
CiteSeer	HeuristicAttack	$71.65 \pm 0.61$	$65.90 \pm 1.18$	$70.64 \pm 1.11$
	Mettack	$71.65 \pm 0.61$	$62.00 \pm 0.69$	$68.82 \pm 0.15$

In contrast, HeuristicAttack, with its superior scalability and more consistent performance across validation conditions, is a more flexible and reliable choice for evaluation.

#### D.4.2 WHY NOT TEXTATTACK?

**Introduction of Text Adversarial Attack.** While our main approach employs LLM-based text generation for adversarial attacks, one may be concerned about the unnoticeability of such substitutions. To address this limitation, we conduct a comprehensive analysis using established NLP adversarial attack methods that explicitly optimize for imperceptibility. Specifically, we employ the TextAttack library and select TextFooler as our primary attack method, including BAE, PWWS, and HotFlip, in preliminary evaluations.

TextFooler operates by strategically perturbing individual words within input sentences to generate semantically equivalent yet syntactically modified text. The method prioritizes semantic preservation while introducing subtle lexical modifications, rendering the perturbations challenging to detect for both human evaluators and automated systems. This characteristic makes TextFooler particularly suitable when imperceptibility constitutes a critical requirement.

*Experimental Configuration.* We configure TextFooler with MiniLM embeddings and utilize default parameters from the TextAttack library (Morris et al., 2020). The victim model is set as GCN. Our evaluation employs a perturbation rate of 0.4 across all experiments. The experiment environment is set as the inductive evasion setting.

*Key Findings.* Our empirical analysis reveals a critical dependency between attack effectiveness and the alignment of embedding representations used in both the attack generation and target model defense mechanisms. As shown in Table 5, when the attack embeddings (MiniLM) match those employed by the target model (MiniLM), TextFooler demonstrates substantially degraded performance across all evaluated GNN architectures, with a notable performance drop. However, as shown in Tables 6 and 7, when embedding misalignment occurs—specifically when target models utilize different embedding schemes such as BoW or RoBERTa, the attack effectiveness diminishes considerably.

These results provide compelling evidence that the text adversarial attack still overfits the surrogate model and the embedding type. To ensure an effective attack strength consistently, we use the LLM-based text attack that generally degrades the performance of all backbones with all encoders.

Table 5: TextFooler attack results, MiniLM embedding for the defender. Bold indicates best performance, underline indicates second best.

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History
GCN	$57.32 \pm 2.84$	$50.78 \pm 2.14$	$63.32 \pm 1.19$	<b><math>74.26 \pm 13.51</math></b>	$52.85 \pm 1.89$	$53.80 \pm 0.49$	$84.47 \pm 0.40$
GAT	<b><math>64.88 \pm 7.50</math></b>	$41.43 \pm 2.29$	<u><math>66.59 \pm 5.63</math></u>	$51.86 \pm 22.41$	$52.38 \pm 4.75$	$53.07 \pm 1.44$	$83.82 \pm 0.60$
APPNP	$53.75 \pm 1.22$	$52.40 \pm 1.67$	$61.94 \pm 0.88$	$60.99 \pm 15.99$	$55.67 \pm 0.83$	$46.40 \pm 0.44$	$84.80 \pm 0.48$
GPRGNN	$50.62 \pm 0.43$	$44.98 \pm 0.68$	$65.68 \pm 1.10$	$59.02 \pm 16.72$	$49.54 \pm 5.39$	$45.54 \pm 0.79$	$82.96 \pm 0.66$
RobustGCN	<b><math>76.32 \pm 3.28</math></b>	<b><math>68.34 \pm 0.80</math></b>	<b><math>72.91 \pm 1.10</math></b>	$64.67 \pm 0.61$	$62.13 \pm 0.47$	<b><math>57.69 \pm 0.83</math></b>	<b><math>84.89 \pm 0.44</math></b>
NoisyGCN	$59.35 \pm 1.61$	$53.24 \pm 1.09$	$63.78 \pm 0.76$	<u><math>64.15 \pm 13.95</math></u>	$50.24 \pm 1.82$	<u><math>54.03 \pm 0.79</math></u>	$84.55 \pm 0.34$
GRAND	$60.02 \pm 1.94$	<u><math>66.93 \pm 1.12</math></u>	$65.35 \pm 0.56$	$49.55 \pm 0.92$	<b><math>63.76 \pm 0.35</math></b>	$51.08 \pm 0.69$	$83.03 \pm 0.52$
EvenNet	$63.22 \pm 2.30$	$59.77 \pm 0.27$	$65.91 \pm 0.97$	$62.95 \pm 15.51$	$54.29 \pm 2.65$	$47.83 \pm 1.58$	$84.73 \pm 0.56$
GNNGuard	$49.57 \pm 0.98$	$46.13 \pm 1.40$	$66.45 \pm 0.32$	$59.48 \pm 15.24$	$42.12 \pm 1.44$	$44.98 \pm 3.69$	$82.91 \pm 0.35$

Table 6: TextFooler attack results, BoW embedding for the defender. Bold indicates best performance, underline indicates second best.

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History
GCN	$85.30 \pm 1.61$	$72.99 \pm 1.93$	$86.13 \pm 0.30$	$81.52 \pm 0.24$	$63.11 \pm 0.70$	$60.54 \pm 0.93$	$81.79 \pm 0.11$
GAT	<b><math>86.04 \pm 1.91</math></b>	<u><math>73.30 \pm 0.94</math></u>	$86.12 \pm 0.20$	$81.33 \pm 0.32$	$64.43 \pm 1.05$	$61.51 \pm 1.21$	<u><math>82.13 \pm 0.60</math></u>
APPNP	$85.73 \pm 2.16$	$70.85 \pm 0.38$	$85.04 \pm 0.11$	$78.46 \pm 1.62$	$63.30 \pm 1.48$	$56.88 \pm 0.95$	$81.35 \pm 0.26$
GPRGNN	$81.49 \pm 1.80$	$69.80 \pm 0.63$	$84.21 \pm 0.43$	$79.18 \pm 1.61$	$63.96 \pm 0.46$	$59.26 \pm 0.97$	$78.28 \pm 0.57$
RobustGCN	<b><math>86.59 \pm 1.52</math></b>	$72.73 \pm 0.13$	$86.39 \pm 0.57$	<u><math>82.41 \pm 0.37</math></u>	<b><math>65.86 \pm 0.40</math></b>	$59.81 \pm 0.39$	$81.48 \pm 0.67$
NoisyGCN	$85.55 \pm 1.28$	$72.36 \pm 1.41$	$86.01 \pm 0.30$	$81.65 \pm 0.35$	$62.82 \pm 1.00$	$61.63 \pm 0.11$	<b><math>82.19 \pm 0.14</math></b>
GRAND	$83.83 \pm 2.18$	<b><math>73.72 \pm 1.61</math></b>	<b><math>87.47 \pm 0.47</math></b>	$80.44 \pm 0.37$	$64.45 \pm 0.25$	<b><math>62.77 \pm 1.99</math></b>	$79.42 \pm 0.68$
EvenNet	$83.70 \pm 2.03$	$71.21 \pm 1.04$	$87.43 \pm 0.29$	<b><math>82.53 \pm 0.35</math></b>	$64.93 \pm 1.39$	$60.06 \pm 0.73$	$81.07 \pm 0.51$
GNNGuard	$80.38 \pm 1.22$	$66.14 \pm 1.23$	$82.94 \pm 0.49$	$69.09 \pm 3.86$	$62.48 \pm 0.46$	$54.75 \pm 0.52$	$78.04 \pm 0.32$

Table 7: TextFooler attack results, RoBerta embedding for the defender. Bold indicates best performance, underline indicates second best.

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History
GCN	<b><math>87.39 \pm 1.00</math></b>	$75.18 \pm 0.87$	$86.35 \pm 0.22$	<u><math>84.39 \pm 0.46</math></u>	$66.06 \pm 0.68$	<b><math>66.41 \pm 0.38</math></b>	$85.01 \pm 0.30$
GAT	$86.84 \pm 1.28$	<u><math>75.60 \pm 0.70</math></u>	$87.15 \pm 0.23$	<u><math>83.80 \pm 0.82</math></u>	$67.09 \pm 0.75$	$63.92 \pm 0.31$	$84.59 \pm 0.54$
APPNP	$79.40 \pm 1.39$	$73.35 \pm 0.92$	$88.21 \pm 0.30$	$80.56 \pm 4.05$	$64.27 \pm 1.16$	$57.39 \pm 0.16$	<u><math>85.74 \pm 0.38</math></u>
GPRGNN	$82.41 \pm 0.68$	$70.53 \pm 1.63$	$85.21 \pm 0.29$	$79.51 \pm 4.44$	$62.24 \pm 1.49$	$54.89 \pm 1.44$	$84.65 \pm 0.62$
RobustGCN	<b><math>86.96 \pm 1.99</math></b>	$74.03 \pm 0.32$	$86.52 \pm 0.09$	$83.28 \pm 0.59$	<b><math>67.39 \pm 0.17</math></b>	$59.42 \pm 0.79$	$84.57 \pm 0.44$
NoisyGCN	$86.90 \pm 0.90$	$75.39 \pm 0.80$	$86.35 \pm 0.28$	$83.75 \pm 1.14$	$66.48 \pm 0.75$	<b><math>66.52 \pm 0.42</math></b>	$85.04 \pm 0.19$
GRAND	$86.10 \pm 1.21$	<b><math>77.27 \pm 0.34</math></b>	<b><math>89.78 \pm 0.35</math></b>	$83.13 \pm 0.85$	$66.49 \pm 0.41$	$63.53 \pm 1.27$	<b><math>85.92 \pm 0.41</math></b>
EvenNet	$83.09 \pm 1.55$	$74.19 \pm 0.64$	<u><math>89.45 \pm 0.57</math></u>	<b><math>84.91 \pm 1.63</math></b>	$65.42 \pm 0.79$	$59.59 \pm 0.36$	$85.62 \pm 0.57$
GNNGuard	$71.40 \pm 1.74$	$68.65 \pm 0.46$	$82.27 \pm 0.82$	$75.48 \pm 5.68$	$60.58 \pm 1.71$	$54.14 \pm 1.26$	$84.64 \pm 0.52$

## E DETAILS OF DEFENSE METHODS

### E.1 INTRODUCTION OF DEFENSE MODEL

For GNNs and RGNNs, we have the following methods as baselines.

#### Spatial/Message Passing Models

1. **GCN** (Kipf & Welling, 2017): Graph Convolutional Network using localized spectral convolution with Chebyshev polynomials.
2. **GAT** (Velickovic et al., 2018): Graph Attention Network employing multi-head attention mechanisms for adaptive neighborhood aggregation.

#### Spectral Models

1. **APPNP** (Klicpera et al., 2019): Approximate Personalized Propagation of Neural Predictions, combining neural predictions with personalized PageRank.
2. **GPRGNN** (Chien et al., 2021): Generalized PageRank Graph Neural Network with learnable graph filter coefficients.
3. **EvenNet** (Lei et al., 2022): Spectral-based defense using even convolution networks with teleportation mechanisms.

#### Robust Training Methods

1. **GRAND** (Feng et al., 2020): Graph Random Neural Networks with consistency regularization using random propagation and DropNode.
2. **NoisyGCN** (Ennadir et al., 2024): GCN with feature noise injection during training to improve robustness.
3. **GPRGNN-AT** (Gosch et al., 2023): GPRGNN with adversarial training. Results are left to Appendix K.

#### Probabilistic Methods

1. **RobustGCN** (Zhu et al., 2019): Robust Graph Convolutional Network with Gaussian-based attention and variance-based message passing.

#### Similarity-based Methods

1. **GNNGuard** (Zhang & Zitnik, 2020): Attention-based neighborhood filtering using cosine similarity thresholds to detect and mitigate adversarial edges.
2. **ElasticGNN** (Liu et al., 2021): Elastic message passing with  $L_1/L_2$  regularization for handling graph heterophily.
3. **SoftMedianGDC** (Geisler et al., 2021): Soft median aggregation with Gaussian Diffusion Convolution and temperature control.
3. **RUNG** (Hou et al., 2024): Robust Graph Neural Networks with uncertainty quantification and Laplacian smoothing.

#### Other Architectural Improvements

1. **GCORN** (Abbahaddou et al., 2024): Higher-order Graph Convolutional Networks with polynomial filters and weight regularization.

### Unsupervised Structure Cleaning

1. **Jaccard-GCN** (Wu et al., 2019): Preprocesses graphs by removing edges with low Jaccard similarity between node features.
2. **Cosine-GCN** (Mujkanovic et al., 2022): Edge filtering based on cosine similarity thresholds between node feature vectors.

### Supervised Structure Learning

1. **ProGNN** (Jin et al., 2020): Joint optimization of graph structure and GNN parameters with sparsity and smoothness constraints.
2. **Stable** (Li et al., 2022b): An unsupervised pipeline that optimizes graph structure by learning edge weights using a metric function combining node feature and structure information. It employs Cosine and Jaccard similarity with learnable thresholds to filter adversarial edges.
3. **GOOD-AT**: It trains an ensemble of OOD detectors using adversarial edges as supervision signals to learn which edges are anomalous, then removes detected perturbations to recover a cleaner graph structure. Results are left to Appendix K.
4. **GPR-GAE**: It learns to reconstruct clean graph structure via a self-supervised graph autoencoder with multiple GPR filters, using the original graph’s reconstruction loss as implicit supervision for structure learning. Results are left to Appendix K.

### GraphLLM Defenses

1. **GraphGPT** (Tang et al., 2024): Graph-text alignment model using contrastive learning between graph embeddings and text representations.
2. **LLaGA** (Chen et al., 2024): Large Language and Graph Assistant that aligns graph structural information with language model representations through multi-modal learning.
3. **SFT with Neighbors** (Wang et al., 2025b): Supervised Fine-Tuning approach that incorporates neighborhood information into LLM prompts for enhanced graph understanding.

## E.2 CONFIGURATION AND HYPERPARAMETERS

This sub-section details configurations and hyperparameters for both GNN-based defenses and GraphLLM methods used in our evaluation framework.

### E.2.1 GNN AND RGNNS

**General Settings** All GNN-based defense models share the following general hyperparameters:

- **Learning rate**: 0.01 (consistent across all methods)
- **Weight decay**: Grid search over [0.0, 0.0005]
- **Dropout**: Grid search over [0.5, 0.7] (except model-specific variations)
- **Hidden dimensions**: 128 for small datasets (Cora, CiteSeer, PubMed, Instagram, WikiCS); 256 for large datasets
- **GAT adjustment**: Hidden dimension reduced by factor of 8 due to multi-head attention
- **Training epochs**: Dataset-dependent with early stopping
  - Small datasets (Cora, CiteSeer, Instagram, PubMed, WikiCS): 400 epochs
  - Medium datasets (Computer, Photo, History, Reddit): 600 epochs
  - Large datasets (ArXiv): 1000 epochs
- **Patience**: Dataset-dependent early stopping
  - Small datasets: 100 epochs patience
  - Medium datasets: 200 epochs patience
  - Large datasets: 400 epochs patience

**Specific Hyperparameters.** The following table summarizes the model-specific hyperparameter ranges for GNN-based defense methods. For the additional baselines in Appendix K, we use the default hyperparameters following official implementations.

Model	Parameter	Values
GCN	None	None
GAT	Num_Heads	8
APPNP	Alpha	[0.1, 0.3, 0.7, 0.9]
GPRGNN	Alpha	[0.1, 0.3, 0.7, 0.9]
RobustGCN	None	None
ElasticGNN	Lambda1	[3, 6]
	Lambda2	[3, 6]
GNNGuard	Threshold	[0.3, 0.4, 0.5, 0.6, 0.7]
NoisyGCN	Beta	[0.1, 0.3, 0.5]
GCORN	None	None
ProGNN	Alpha	[0.0005, 0.3]
	Beta	[1.5, 2.5]
Stable	Cosine Threshold	[0.3, 0.5, 0.7]
	Jaccard Threshold	[0.02, 0.03]
	Alpha	[0.1, 0.03, 0.6]
GCN-Jaccard	Jaccard Threshold	[0.03, 0.05, 0.1]
GCN-Cosine	Cosine Threshold	[0.3, 0.4, 0.5, 0.6, 0.7]
GRAND	Dropnode	[0.5]
	Order	[2, 4]
	MLP Input Dropout	[0.5]
	N Samples	[4]
	Reg Consistency	[0.7, 1.0]
	Sharpening Temperature	[0.5]
SoftMedianGDC	Temperature	[0.5, 1.0]
	Teleport Probability	[0.15, 0.25]
	Neighbors	[64]
RUNG	Lambda	[0.7, 0.9]
	Gamma	[1, 3]
EvenNet	K	[10]
	Alpha	[0.1, 0.3, 0.7, 0.9]
	DP Rate	[0.5]

### E.2.2 GRAPHLLMs

**General Configurations** GraphLLM methods employ distinct training configurations optimized for large-scale language model integration:

- **Base LLM:** Mistral-7B (4096-dimensional output) as primary backbone
- **Text encoding:** RoBERTa for feature extraction and alignment
- **Batch processing:** Varies by model complexity (8-64 samples per batch)
- **Learning rates:** Lower rates (1e-4) for stable LLM fine-tuning
- **Weight decay:** 0.05 for regularization
- **Gradient clipping:** Applied to prevent exploding gradients
- **Mixed precision:** Enabled for memory efficiency

**GraphGPT.** For GraphGPT, we adhere to the best hyperparameters in clean datasets provided by (Wu et al., 2025).

**LLaGA.** For LLaGA, we adhere to the best hyperparameters in clean datasets provided by (Wu et al., 2025).

- **Language model embedding:** RoBERTa
- **Neighborhood template:** HO (Hopfield) encoding

**SFT with Neighbors.** Supervised Fine-Tuning incorporates neighborhood-aware prompting with LoRA optimization:

- **Maximum neighbors:** maximum\_neighbor=6 for context window management
- **Neighbor filtering strategy:** Degree-based selection - neighbors are ranked by node degree in descending order, and top-6 highest-degree neighbors are selected for prompt inclusion
- **LoRA configuration:** r=8, alpha=16, dropout=0.1, target\_modules=[q\_proj, v\_proj]
- **Sequence lengths:** max\_txt\_length=128, max\_origin\_txt\_length=128, max\_ans\_length=16
- **Optimization:** AdamW optimizer with gradient accumulation
- **Prompt engineering:** Integrates 1-hop neighbor information with degree-based prioritization for enhanced context understanding

*Neighbor-aware Prompt Template:*

**Node Classification Task**

Question: You are doing node classification task in a citation graph. Given the content of the center node: `{origin_text}` and its neighbor information: `{neighbor_text}`, each node represents a paper and the relationship represents the citation relationship between papers, we need to classify the center node into 7 classes: `{classes}`. Please tell me which class the center node belongs to?  
Answer only the class name without any other words.

Answer:

### E.2.3 GRAPHLLMs WITH NOISY TRAINING

This subsection covers noise injection strategies across different GraphLLM architectures in Section

**GraphGPT with Noise.** GraphGPT incorporates noise injection through graph-level modifications using the noise\_utils framework. Supports only the structural “noise” variant. Noise is applied globally to the entire graph structure before training. Configurations follow the clean dataset parameters from (Wu et al., 2025) with additional noise strategies applied during contrastive learning. The perturb ratio is set to 10%.

GraphGPT-noisetxt and GraphGPT-noisefull are excluded because the noise injected hurts clean performance too much, as shown in Table 9.

Table 9: Performance comparison of GraphGPT variants across under the inductive setting on clean datasets.

Method	Cora	CiteSeer	PubMed	WikiCS
GraphGPT	$81.06 \pm 2.33$	$74.35 \pm 2.51$	$94.14 \pm 0.23$	$82.31 \pm 1.31$
GraphGPT-noise	$80.63 \pm 2.24$	$74.56 \pm 3.07$	$94.09 \pm 0.33$	$82.32 \pm 1.97$
GraphGPT-noisetxt	$66.79 \pm 3.84$	$57.37 \pm 5.20$	$86.15 \pm 1.88$	$64.56 \pm 2.94$

**LLaGA with Noise.** LLaGA applies noise injection through graph-level modifications using the noise\_utils framework. The noise strategies modify the global graph structure or replace text content across the entire graph before multi-modal learning. Noise integration occurs with the RoBERTa language model embedding and HO (Hopfield) neighborhood template encoding. Base configurations identical to clean LLaGA training. The perturb ratio is set to 10%.

**SFT with Noise Variants.** SFT implements three distinct noise injection strategies during training:

*Noise (Structure):* Injection strategy targets high-degree nodes with 10% probability, adding random unconnected nodes as fake neighbors.

*NoiseTxt (Text):* Injection strategy replaces text content of 10% high-degree nodes with text from different-class nodes.

*Noisefull (Both):* Injection strategy combines both structural and text noise - applies both neighbor injection (10% probability) and text replacement (10% of high-degree nodes) simultaneously.

- **Injection ratio:** 10% probability for structural noise, 10% of high-degree nodes for text replacement
- **Target selection:** High-degree nodes (degree > average degree)
- **Replacement strategy:** Random selection from unconnected nodes (structural) or different-class nodes (text)
- **Training augmentation:** Applied only during the training phase, inference uses clean data
- **Base configurations:** Identical to SFT with Neighbors for all other parameters

*SFT Noise Training Strategy (Sample-Level):* Unlike GraphGPT/LLaGA which apply graph-level noise modifications, SFT applies noise injection at the sample level during data preparation. For structural noise, 10% probability of adding random unconnected nodes as fake neighbors to high-degree nodes during neighbor selection for each training sample. For text noise, 10% of high-degree nodes have their text content replaced with text from nodes of different classes during individual sample creation. NoiseFull combines both strategies with independent application - both structural neighbor injection and text replacement occur simultaneously for each sample.

#### E.2.4 SFT WITH SIMILARITY CONSTRAINTS

**SFT-simp.** The similarity variant teaches the model to selectively use neighbor information based on similarity constraints through prompt engineering:

- **Training:** Standard neighbor-based training (identical to SFT with Neighbors)
- **Inference strategy:** Prompt-based similarity awareness
- **Similarity criteria:** Text content and label consistency

*Similarity-Aware Prompt Template:*

**Node Classification with Similarity Constraints**

Question: You are doing node classification task in a citation graph. Given the content of the center node: `{origin_text}` and its neighbor information: `{neighbor_text}`, each node represents a paper and the relationship represents the citation relationship between papers, we need to classify the center node into `{num_classes}` classes: `{classes}`. Consider neighbor information for classification ONLY when: 1) neighbors are similar to the center node, or 2) neighbors are similar to each other. Similarity can be based on text content or label consistency. Otherwise, ignore neighbor information. Please tell me which class the center node belongs to? Answer only the class name without any other words.

Answer:

#### E.2.5 SFT-AUTO

**Pipeline.** The auto variant implements comprehensive attack detection and recovery through multi-stage inference with specialized prompt templates. The pseudo-code of the algorithm is provided in Algorithm 1, and the related prompts are listed below.

*Attack Detection Prompt Template:*

**Node Classification with Attack Detection**

Question: You are doing node classification in a citation graph. Given the content of the center node: `{origin_text}` and its neighbor information: `{neighbor_text}`, classify the center node into `{num_classes}` classes: `{classes}` or “text\_attacked”. The center node may be attacked. If its class is unclear or differs from most of its neighbors, classify it as “text\_attacked” instead. Please tell me which class the center node belongs to? Answer only the class name without any other words.

Answer:

*Recovery Prompt Template (Neighbor-Only):*

**Recovery Using Neighbor Information Only**

Question: You are doing node classification task in a citation graph. Based only on the neighbor information: `{neighbor_text}`, each node represents a paper and the relationship represents the citation relationship between papers, we need to classify the center node into `{num_classes}` classes: `{classes}`. Please predict the class of the center node using only neighbor information. Answer only the class name without any other words.

Answer:

**Algorithm 1:** Auto Variant Framework for Adversarial Attack Detection and Recovery

**Input:** Graph data  $G = (V, E, X)$ , training nodes  $V_{\text{train}}$ , test nodes  $V_{\text{test}}$ , similarity threshold  $\tau$

**Output:** Trained model  $\mathcal{M}$ , final predictions  $\mathbf{y}_{\text{pred}}$

```

1 Training Phase:
2  $r \leftarrow \min\left(\frac{1}{|\mathcal{C}|}, 0.15\right)$  // Adaptive attack ratio
3  $\mathcal{S}_{\text{normal}} \leftarrow \text{PREPARENORMALSAMPLES}(V_{\text{train}});$ 
4  $\mathcal{S}_{\text{attack}} \leftarrow \text{GENERATETEXTATTACKSAMPLES}(V_{\text{train}}, r);$ 
5  $\mathcal{S}_{\text{recovery}} \leftarrow \text{GENERATERECOVERYSAMPLES}(\mathcal{S}_{\text{attack}});$ 
6  $\mathcal{S}_{\text{all}} \leftarrow \mathcal{S}_{\text{normal}} \cup \mathcal{S}_{\text{attack}} \cup \mathcal{S}_{\text{recovery}};$ 
7  $\mathcal{M} \leftarrow \text{TRAINMODEL}(\mathcal{S}_{\text{all}});$ 

8 Inference Phase:
9  $\mathcal{I}_{\text{text}} \leftarrow \emptyset, \mathcal{I}_{\text{struct}} \leftarrow \emptyset;$ 
    // Stage 1: Attack Detection
10 foreach  $v_i \in V_{\text{test}}$  do
11   if  $\text{LOWSIMILARITYNEIGHBORS}(v_i, \tau) \geq |\mathcal{N}_i|/2$  then
12      $\mathcal{I}_{\text{struct}} \leftarrow \mathcal{I}_{\text{struct}} \cup \{i\};$ 
13   if  $\mathcal{M}(v_i) = \text{"text\_attacked"}$  then
14      $\mathcal{I}_{\text{text}} \leftarrow \mathcal{I}_{\text{text}} \cup \{i\};$ 
    // Stage 2: Recovery and Final Prediction
15 foreach  $v_i \in V_{\text{test}}$  do
16   if  $i \in \mathcal{I}_{\text{text}}$  then
17      $\mathbf{y}_{\text{pred}}[i] \leftarrow \mathcal{M}(\text{neighbors\_only}(v_i));$ 
18   else if  $i \in \mathcal{I}_{\text{struct}}$  then
19      $\mathbf{y}_{\text{pred}}[i] \leftarrow \mathcal{M}(v_i, \text{filtered\_neighbors}(v_i, \tau));$ 
20   else
21      $\mathbf{y}_{\text{pred}}[i] \leftarrow \mathcal{M}(v_i, \mathcal{N}_i);$ 
22 return  $\mathcal{M}, \mathbf{y}_{\text{pred}}$ 

```

## F RESULTS AGAINST WTGIA

In this section, we present results against Text-level GIA and WTGIA attacks.

We introduced the WTGIA methodology and basic settings in Section D.3. Specifically, we evaluate on three datasets: Cora, CiteSeer, and PubMed, following the original paper. We present results for two perturbation rates: `ptb_rate = 0.2` and `ptb_rate = 0.4`.

At  $\text{ptb\_rate} = 0.2$ , WTGIA introduces the same number of edges as structural evasion attacks. Considering that in GMA, each edge modification affects two nodes in the original graph, we also conduct experiments at  $\text{ptb\_rate} = 0.4$ , where the number of affected edges approximately equals that of GMA. By aligning the attack budgets between GMA and Text-level GIA, we can study the attack strength of Text-level GIA compared to GMA.

Following our other experiments, we evaluate both GNN and RGNN models using three embedding types: BoW, MiniLM, and RoBERTa. Note that WTGIA inherently uses BoW as the victim text encoder. In this experiment, we investigate how different text encoders affect defense performance against WTGIA.

Table 10: WTGIA Attacked Test Accuracy (Perturb ratio being 20%)

Method	Cora			CiteSeer			PubMed		
	BoW	MiniLM	RoBERTa	BoW	MiniLM	RoBERTa	BoW	MiniLM	RoBERTa
GCN	59.29	83.46	85.36	49.22	71.73	74.03	77.49	84.65	83.95
GAT	57.56	80.87	82.60	45.61	66.30	67.29	74.02	74.99	82.21
APPNP	60.95	84.13	<u>86.41</u>	65.62	<u>76.02</u>	74.66	85.07	89.03	91.22
GCORN	53.38	83.83	81.67	47.49	73.20	72.47	61.15	79.74	80.92
GPRGNN	69.43	83.76	84.93	64.63	73.56	74.56	83.99	89.44	90.78
GRAND	59.78	84.62	<b>86.78</b>	56.22	75.76	<b>76.38</b>	80.03	86.49	89.47
EvenNet	78.78	82.53	82.60	68.86	73.77	73.41	83.64	86.66	89.19
ElasticGNN	60.02	84.50	85.42	54.49	73.88	73.30	78.25	79.68	85.86
RobustGCN	78.91	83.64	85.61	68.34	73.77	73.98	81.63	79.44	82.06
GNNGuard	67.59	82.90	83.15	65.46	75.24	73.72	82.40	88.95	90.29
SoftmedianGDC	80.87	84.87	83.70	71.79	75.24	74.71	86.32	89.54	91.20
NoisyGCN	63.78	83.15	85.06	49.63	72.10	73.30	78.26	85.24	86.03
RUNG	79.52	83.27	83.95	70.27	74.03	73.56	85.68	89.64	90.99
GraphGPT		74.97			72.83			92.84	
LLaGA		83.21			72.36			89.76	
SFT-neighbor		79.89			73.51			<b>94.74</b>	

Table 11: WTGIA Attacked Test Accuracy (Perturb ratio being 40%)

Method	Cora			CiteSeer			PubMed		
	BoW	MiniLM	RoBERTa	BoW	MiniLM	RoBERTa	BoW	MiniLM	RoBERTa
GCN	47.91	82.16	84.99	33.28	70.01	72.73	75.77	79.54	79.09
GAT	47.23	76.32	80.44	27.90	60.92	62.38	66.59	63.28	72.74
APPNP	50.55	84.01	85.67	60.29	<b>75.39</b>	74.61	86.07	88.88	91.18
GCORN	32.10	82.23	81.00	26.12	70.32	67.08	53.20	77.50	77.49
GPRGNN	64.45	82.78	83.70	52.87	73.20	72.94	84.04	89.01	90.64
GRAND	48.03	84.44	<b>86.10</b>	44.36	74.97	74.29	76.36	86.16	88.90
EvenNet	73.37	81.06	81.49	66.25	73.51	73.30	77.50	81.52	86.51
ElasticGNN	46.19	83.27	83.64	37.57	72.52	71.89	75.57	70.76	80.49
RobustGCN	75.03	81.80	<u>85.85</u>	62.12	73.35	72.99	75.97	70.62	75.52
GNNGuard	61.32	82.78	83.76	61.86	<u>75.34</u>	73.41	81.11	89.05	90.24
SoftmedianGDC	78.97	84.38	83.95	70.79	74.92	74.61	86.17	89.77	90.99
NoisyGCN	52.46	82.16	83.76	34.12	71.26	71.89	75.22	81.03	81.67
RUNG	79.83	82.66	84.19	68.13	73.46	74.76	85.81	89.77	91.11
GraphGPT		71.16			70.22			92.29	
LLaGA		81.86			68.23			89.32	
SFT-neighbor		77.80			73.20			<b>94.62</b>	

The experimental results are presented in Table 10 and Table 11. We have the following discoveries: **Cross-Modal Attack Transferability**: WTGIA exhibits significantly stronger attack performance against BoW-based models compared to advanced text encoders like MiniLM and RoBERTa, even under a strong budget (40%). This pronounced performance gap highlights the limited transferability of text-level GIAs across different embeddings, suggesting that WTGIA overfits to the victim’s specific encoder (BoW). In contrast, structural attacks display slightly more consistent degradation across text representations, underscoring the modality-specific nature of text-level perturbations.

**Vulnerability of Text-Aware Models**: Text-level perturbations disproportionately impact models reliant on textual features, particularly LLM-based approaches like SFT-neighbor. On the Cora

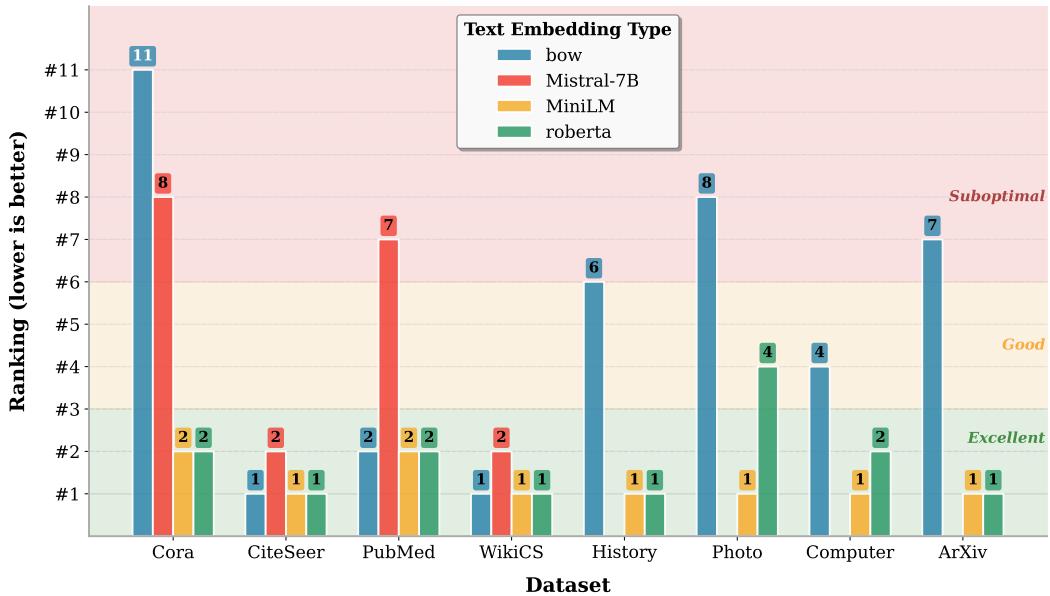


Figure 7: GNNGuard performance ranking across different text embeddings under structure attacks. The bar chart displays ranking positions (lower is better) for each embedding type across eight benchmark datasets in the inductive setting with a perturbation rate of 0.2.

dataset, SFT-neighbor maintains competitive accuracy (82%) under structural attacks but suffers greater degradation under WTGIA.

## G WHY GNNGuard EXCELS: UNDERSTANDING AND IMPROVING

In the main text, we observe that GNNGuard achieves remarkable performance improvements. This raises a fundamental question: what drives GNNGuard’s exceptional effectiveness? In this section, we provide a comprehensive analysis to dissect the underlying mechanisms.

### G.1 RESULTS OF GNNGuard UNDER DIFFERENT TEXT ENCODERS

In Figure 7, we plot the performance of GNNGuard against structure evasion attacks using different text encoders. We exclude the results on Instagram and Reddit because the performance differences among all methods on these datasets are not significant. In the ranking, we remove the GraphLLM methods because they do not necessarily depend on embeddings. The results reveal substantial variations in defensive effectiveness depending on the embedding choice. In previous works (Zhang & Zitnik, 2020; Lei et al., 2022; Mujkanovic et al., 2022; Hou et al., 2024), GNNGuard is evaluated using embeddings like BoW and TF-IDF, where its ranking is indeed low as shown in the table. However, when switching to advanced language model embeddings like MiniLM and RoBERTa, its ranking improves significantly. For instance, on Cora, when using embeddings like BoW and Mistral-7B, its ranking falls within the suboptimal region, but when employing MiniLM and RoBERTa, it demonstrates improved ranking relative to other RGNNs and GNNs, indicating that text encoders significantly influence GNNGuard’s ranking.

### G.2 DIFFERENCE BETWEEN TEXT-ENCODERS

To understand why the text encoders significantly affect the performance of GNNGuard, we systematically evaluate the effectiveness of different text embedding methods for similarity-based edge filtering, which is essential for its defense mechanism. Our analysis compares BoW, TF-IDF, Mistral-7B, RoBERTa, and MiniLM embeddings across multiple graph datasets to identify which representations best distinguish intra-class from inter-class edges.

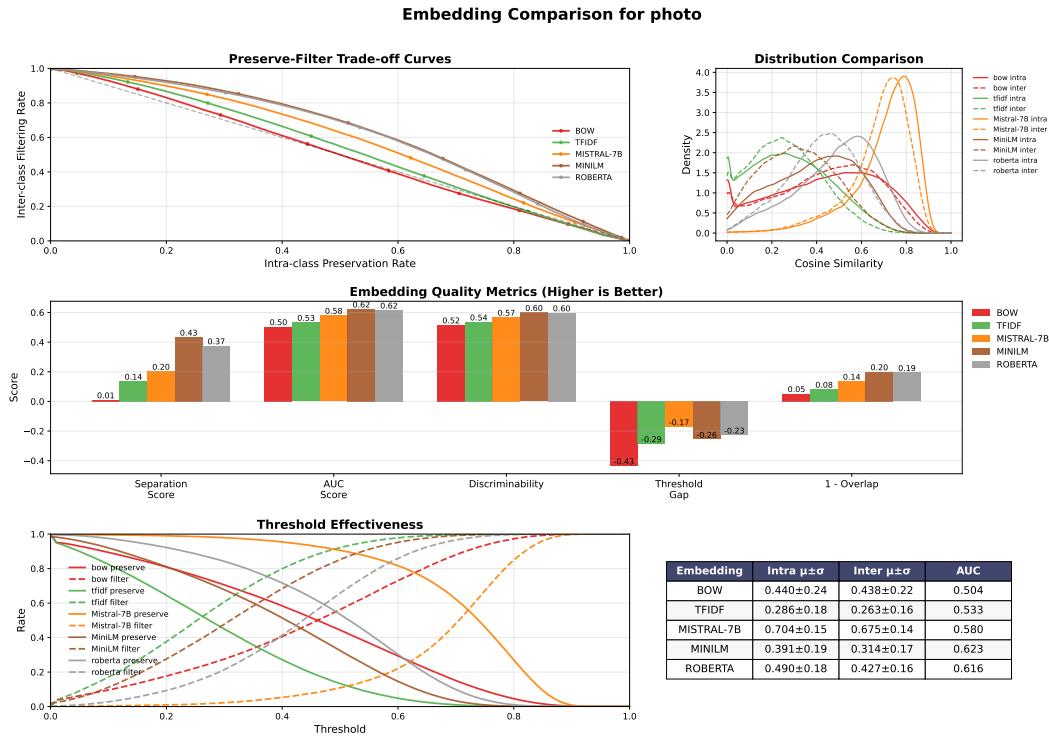


Figure 8: Comprehensive embedding comparison for the Photo dataset. (a) Preserve-filter trade-off curves showing the Pareto frontier between intra-class preservation and inter-class filtering rates; (b) KDE-smoothed similarity distributions revealing the separation between intra-class (solid) and inter-class (dashed) edges; (c) Quantitative quality metrics including separation score, AUC, discriminability, threshold gap, and non-overlap score; (d) Threshold effectiveness curves illustrating how preservation and filtering rates vary with similarity thresholds; (e) Summary statistics table presenting mean similarities and standard deviations for each embedding type.

Figure 8 presents our comprehensive analysis framework using the Photo dataset as an example. The multi-panel visualization reveals critical insights about embedding effectiveness:

- **Preserve-Filter Trade-off (Panel a):** Each curve represents an embedding's ability to simultaneously preserve intra-class edges while filtering inter-class edges across 101 similarity thresholds. The curves are generated by computing, for each threshold  $\tau \in [0, 1]$ , the fraction of intra-class edges with similarity  $\geq \tau$  (x-axis) and inter-class edges with similarity  $< \tau$  (y-axis). Embeddings with curves closer to the upper-left corner exhibit superior discriminative capacity. In the Photo dataset, RoBERTa's curve dominates, achieving 80% inter-class filtering while maintaining 40% intra-class preservation.
- **Similarity Distributions (Panel b):** Kernel density estimation with Gaussian kernels ( $\sigma$  selected via Scott's rule) visualizes the probability density functions of cosine similarities. The aggregation across three random seeds ensures robustness. For Photo, MiniLM, and RoBERTa exhibit clear bimodal separation with intra-class similarities and inter-class differences, while others show substantial overlap with both distributions.
- **Quality Metrics (Panel c):** Five metrics quantify embedding effectiveness:
  - Separation Score: Cohen's  $d = (\mu_{\text{intra}} - \mu_{\text{inter}}) / \sqrt{(\sigma_{\text{intra}}^2 + \sigma_{\text{inter}}^2) / 2}$
  - AUC Score: Area under the ROC curve, treating edge classification as a binary prediction task
  - Discriminability:  $\max_{\tau} [(P(\text{sim}_{\text{intra}} \geq \tau) + P(\text{sim}_{\text{inter}} < \tau)) / 2]$
  - Threshold Gap:  $Q_{20}(\text{sim}_{\text{intra}}) - Q_{80}(\text{sim}_{\text{inter}})$  where  $Q_p$  denotes percentile
  - Non-overlap:  $1 - \sum_i \min(h_{\text{intra}}(i), h_{\text{inter}}(i)) \Delta x$  using 50-bin histograms

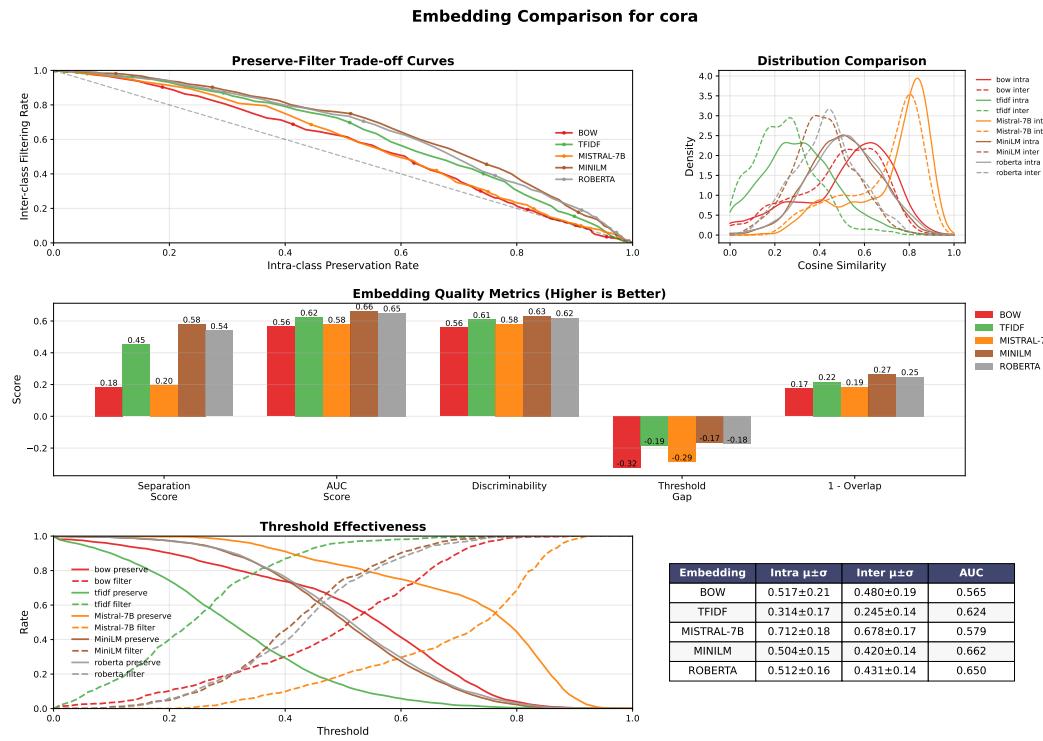


Figure 9: Comprehensive embedding comparison for the Cora dataset. (a) Preserve-Filter Trade-off Curves showing the Pareto frontier between intra-class preservation and inter-class filtering rates; (b) KDE-smoothed similarity distributions revealing the separation between intra-class (solid) and inter-class (dashed) edges; (c) Quantitative quality metrics including separation score, AUC, discriminability, threshold gap, and non-overlap score; (d) Threshold Effectiveness curves illustrating how preservation and filtering rates vary with similarity thresholds; (e) Summary statistics table presenting mean similarities and standard deviations for each embedding type.

- **Threshold Effectiveness (Panel d):** This analysis reveals the operational characteristics of each embedding. The solid lines show  $P(\text{sim}_{\text{intra}} \geq \tau)$  while dashed lines show  $P(\text{sim}_{\text{inter}} < \tau)$  as functions of threshold  $\tau$ . The vertical separation between paired curves indicates discriminative power at each threshold.

#### • Statistical Summary (Panel e):

This analysis framework demonstrates that contextual embeddings (RoBERTa, MiniLM) provide superior edge discrimination compared to sparse representations. While all GNNs and RGNs benefit from better embeddings in the classification task, GNNGuard enjoys more protection against structural attacks with better representations due to the better distinguishability of a better embedding. Another example on the Cora dataset is provided in Figure 9.

### G.3 GUARDUAL: ROBUSTNESS AGAINST TEST-TIME STRUCTURAL ATTACKS

Despite achieving promising results with advanced embeddings, GNNGuard exhibits several limitations in practice. First, the current threshold selection relies on validation set performance, which leads to overly conservative thresholds due to the inherent performance-robustness trade-off on clean validation data. Second, in real-world scenarios where training data integrity can be more readily ensured, applying aggressive thresholds during training risks excessive edge filtering, thereby degrading model performance on benign graphs.

To address these limitations, we introduce Guardual, an adaptive defense mechanism that employs dual similarity thresholds to balance training stability with robust defense capabilities. This method

uses the optimal threshold for preserving benign graph structure during training, which differs from the threshold needed for effective adversarial filtering during inference.

### G.3.1 DUAL-THRESHOLD DESIGN

Guardual employs two complementary thresholds optimized for different phases of model deployment:

- **Conservative Threshold (Training):** This threshold prioritizes preserving intra-class edges to maintain graph connectivity and training stability. Computed using a weighted objective function:  $\text{score}_{\text{conservative}} = 0.7 \cdot P_{\text{preserve}} + 0.3 \cdot P_{\text{filter}}$ , where  $P_{\text{preserve}}$  represents the fraction of intra-class edges retained and  $P_{\text{filter}}$  denotes the fraction of inter-class edges removed. The 70-30 weighting ensures that sufficient graph structure remains intact during the learning phase, preventing performance degradation from excessive edge pruning.
- **Balanced Threshold (Testing):** During inference, the model switches to a balanced threshold that equally weights preservation and filtering:  $\text{score}_{\text{balanced}} = 0.5 \cdot P_{\text{preserve}} + 0.5 \cdot P_{\text{filter}}$ . This equal weighting provides stronger defense against adversarial edges while accepting slightly reduced intra-class preservation, as gradient flow is no longer a concern during evaluation.

Given specific embeddings, we can pre-compute all scores and obtain training/test thresholds. The thresholds for Guardual using RoBERTa embeddings are listed in Table 12.

Table 12: Dual thresholds employed by Guardual across different datasets. Conservative thresholds prioritize edge preservation during training, while balanced thresholds enhance adversarial filtering during testing.

Dataset	Training Threshold (Conservative)	Testing Threshold (Balanced)
Cora	0.250	0.503
CiteSeer	0.320	0.580
PubMed	0.360	0.633
WikiCS	0.283	0.433
Instagram	0.000	0.471
Reddit	0.000	0.177
History	0.260	0.457
Photo	0.000	0.476
Computer	0.000	0.457
ArXiv	0.320	0.540

### G.3.2 RESULTS OF GUARDUAL

The results are presented in Figure 10. We can see that, although it removes a hyperparameter compared to GNNGuard, Guardual achieves general improvement. In Computer and Photo datasets (highlighted in red), Guardual demonstrates the most significant enhancements, with improvements of +8.23% and +10.74% respectively. As shown in the result performance, it become the most robust RGNN against structural evasion attacks, despite its simple and effective design.

## H ADAPTIVE ATTACKS

### H.1 ADAPTIVE ATTACKS AGAINST GNNGUARD

Following Mujkanovic et al. (2022), we investigate adaptive attacks targeting GNNGuard, leveraging its robust performance against structural attacks. We employ a modified PGD attack Xu et al. (2019), termed PGDGuard, which restricts the attack search space to edges with similarity exceeding a threshold  $\epsilon$ . All edge additions or removals satisfy this similarity constraint. We fix the attack embedding to RoBERTa and evaluate PGDGuard across  $\epsilon$  values of 0.0, 0.3, 0.5, and 0.7, assessing the performance of GNN and RGNN models under these conditions. Results are presented in Figure 11.

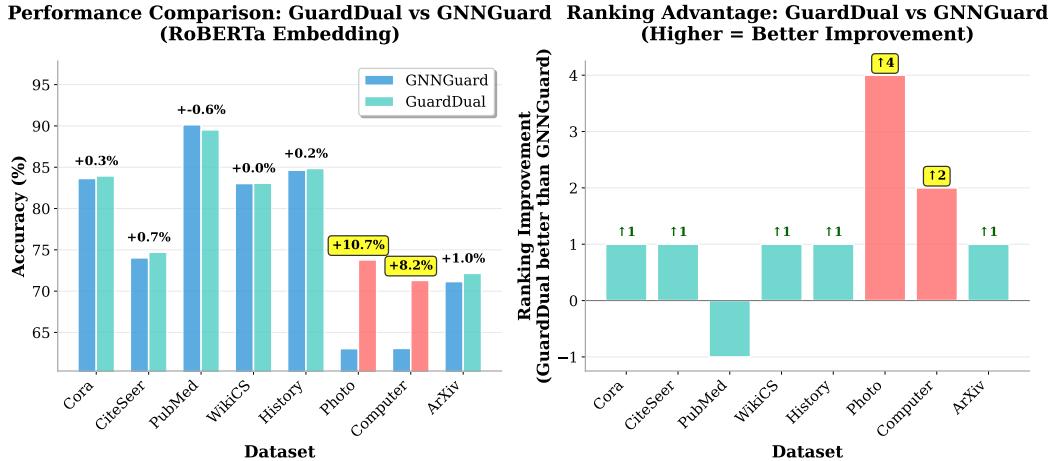


Figure 10: Comprehensive comparison between GuardDual and GNNGuard under RoBERTa embedding for structure attack defense. Left subplot shows absolute accuracy performance with GuardDual consistently outperforming GNNGuard across most datasets. Right subplot displays ranking improvements, where positive values indicate GuardDual’s superior competitive position.

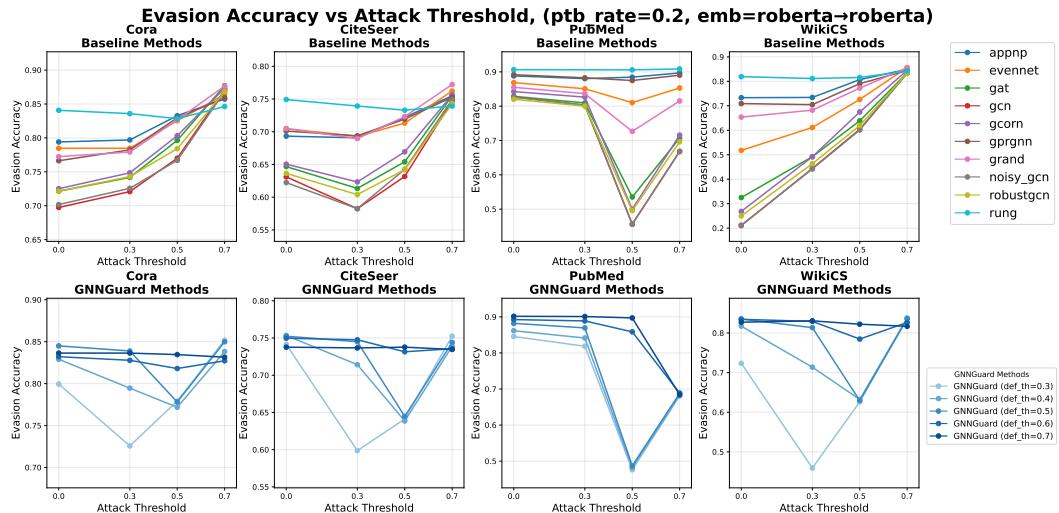


Figure 11: Evasion accuracy vs attack threshold for different defense methods with perturbation rate 0.2 and embedding types roberta.

Key observations are summarized as follows:

- **U-Shaped Performance Curve for GNNGuard:** GNNGuard exhibits a U-shaped trend in evasion accuracy with respect to PGDGuard’s attack threshold  $\epsilon$ . When  $\epsilon$  significantly deviates from GNNGuard’s filtering threshold (either lower or higher), the attack’s effectiveness diminishes. However, when  $\epsilon$  closely aligns with GNNGuard’s threshold, the attack achieves optimal evasion, indicating a trade-off: at lower  $\epsilon$ , PGDGuard’s perturbations lack sufficient potency, while at higher  $\epsilon$ , GNNGuard’s filtering mechanism effectively mitigates the attack.
- **Trade-Offs for Other Defense Methods:** While increasing the attack threshold  $\epsilon$  may amplify PGDGuard’s impact on GNNGuard, it generally enhances the performance of other defense methods. On datasets like Cora and WikiCS, attacks tailored to GNNGuard’s threshold tend to weaken against alternative defenses, posing significant challenges for attackers aiming to generalize across diverse defense strategies.

Table 13: Performance (%) for PGD-structure on **GNNs** (GCN, GRAND, RUNG). Rows = attacker embeddings; columns = defender embeddings.

Model / Dataset	Cora			CiteSeer			PubMed			WikiCS		
	BoW	MiniLM	RoBERTa	BoW	MiniLM	RoBERTa	BoW	MiniLM	RoBERTa	BoW	MiniLM	RoBERTa
<i>GCN</i>												
Attacker = BoW	66.11	72.39	73.31	54.86	61.34	62.70	80.12	83.87	82.92	20.39	30.44	31.54
Attacker = MiniLM	70.11	62.98	68.39	54.60	47.13	54.18	82.60	81.57	83.55	33.25	25.37	28.25
Attacker = RoBERTa	71.34	70.85	69.43	63.95	63.74	63.58	82.21	82.29	82.73	25.12	21.50	20.86
<i>GRAND</i>												
Attacker = BoW	71.89	78.60	79.89	64.00	70.22	70.74	82.45	83.88	86.41	48.57	66.75	71.86
Attacker = MiniLM	75.28	70.91	77.06	64.79	64.99	66.88	84.02	83.43	86.03	64.96	53.57	67.01
Attacker = RoBERTa	75.40	76.45	76.51	68.03	70.43	70.06	84.54	84.25	85.77	57.21	61.13	65.36
<i>RUNG</i>												
Attacker = BoW	78.04	83.64	84.56	67.97	73.46	73.30	85.70	89.78	90.72	75.00	79.95	82.73
Attacker = MiniLM	78.04	83.70	83.83	67.92	73.35	73.93	85.87	89.53	90.61	76.18	80.26	82.53
Attacker = RoBERTa	79.09	83.58	84.32	69.04	74.14	73.93	86.06	89.56	90.72	75.94	80.14	82.39

Table 14: Perfomrance (%) for PGD-structure on **LLaGA** and **SFT-Neighbor**. Rows = attacker embeddings; columns = datasets.

Model / Attacker	Cora	CiteSeer	PubMed	WikiCS
<i>LLaGA</i>				
BoW	75.21	66.35	87.38	66.99
RoBERTa	72.88	67.24	86.70	60.47
<i>SFT-Neighbor</i>				
BoW	82.59	74.24	92.29	84.05
RoBERTa	81.24	71.84	<b>94.72</b>	86.19

- **Dataset-Specific Exceptions:** Despite the general trade-off, notable exceptions arise, particularly on PubMed at  $\epsilon = 0.5$ . Here, PGDGuard achieves superior attack performance across all methods. This is attributed to PubMed’s fine-grained classification, where embedding similarities (metric in subplot b of Figure 8) for both intra-class and inter-class nodes cluster around 0.6. Selecting  $\epsilon = 0.5$  effectively targets potentially harmful edges, highlighting the efficacy of dataset-specific, embedding-aware attack strategies.

These findings underscore the nuanced interplay between attack thresholds and defense mechanisms, emphasizing the importance of aligning attack strategies with dataset-specific embedding characteristics to maximize evasion effectiveness.

## H.2 ADAPTIVE ATTACKS ON EMBEDDING TRANSFERABILITY

**Setup.** In the main experiments, we initially used BoW as the victim’s text embedding for structural evasion attacks. To model adaptive attackers and embedding-aware defenders, we vary the text encoder used the victim model in PGD among {BoW, MiniLM, RoBERTa} while the defender may defend with {BoW, MiniLM, RoBERTa}. We evaluate GCN, GRAND, RUNG, LLaGA, and SFT-Neighbor on datasets Cora, CiteSeer, PubMed, and WikiCS.

The results are shown in Table 13 and Table 14. We have the following findings:

- **Embedding Match Helps the Attacker.** Across models, the attack performance is highest when the attacker and defender use the same embedding. If embedding is mismatched, even from advanced to BoW, the attack performance significantly degrades. In fact, when the surrogate text encoder is LM, BoW as the defender’s embedding can be strong.
- **Transfer is Stronger within LM-family.** MiniLM  $\leftrightarrow$  RoBERTa transfers better than BoW  $\leftrightarrow$  LM.
- **Transferability to GraphLLMs Varies among Datasets.** LLaGA, which takes RoBERTa as the text encoder, also suffers more if the attacker uses RoBERTa as the surrogate. However, for SFT, the results vary among datasets. This observation means that the effective text encoder for attackers could depend on the dataset’s characteristics and the victim model.

## I ABLATION STUDY FOR SFT VARIANTS

In this subsection, we conduct an ablation study to analyze the impact of different neighbor selection strategies and prompt templates in our SFT variants. We examine the effectiveness of degree-based neighbor selection versus random selection, and investigate the influence of incorporating label information in the prompting strategy.

### I.1 IMPACT OF DIFFERENT PROMPT TEMPLATES

Due to context length limitations, currently, we feed nodes’ neighbors with the top degree as representatives in the prompt. In this subsection, we conduct an ablation study to examine the impact of randomly selecting neighbors.

#### I.1.1 DEGREE-BASED VS RANDOM NEIGHBOR SELECTION

Table 15: Comparison between degree-based (SFT-neighbor) and random (SFT-rand) neighbor selection strategies across different attack scenarios. **Bolded values** indicate the better performance between the two methods for each dataset and scenario.

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History
<i>Clean Results on Inductive Split</i>							
SFT-neighbor	<b>88.25 ± 1.29</b>	76.17 ± 1.81	<b>95.08 ± 0.51</b>	<b>87.75 ± 0.46</b>	69.24 ± 0.54	66.74 ± 1.06	<b>86.81 ± 0.77</b>
SFT-rand	87.27 ± 2.30	<b>77.27 ± 0.47</b>	95.05 ± 0.20	87.72 ± 0.58	<b>69.96 ± 0.72</b>	<b>68.46 ± 0.09</b>	86.64 ± 0.45
<i>Structure Attack Results on Inductive Split</i>							
SFT-neighbor	<b>82.35 ± 1.94</b>	<b>71.53 ± 1.86</b>	94.73 ± 0.41	<b>86.00 ± 1.40</b>	68.25 ± 0.20	62.29 ± 3.31	<b>85.98 ± 0.51</b>
SFT-rand	79.09 ± 2.82	71.42 ± 0.48	<b>94.79 ± 0.17</b>	85.19 ± 0.68	<b>68.50 ± 0.71</b>	<b>65.23 ± 1.16</b>	84.77 ± 0.83
<i>Text Attack Results on Inductive Split</i>							
SFT-neighbor	75.65 ± 1.33	43.84 ± 2.14	72.27 ± 3.53	<b>51.53 ± 1.24</b>	<b>64.33 ± 1.46</b>	65.52 ± 1.24	64.84 ± 0.98
SFT-rand	<b>78.60 ± 1.78</b>	<b>47.08 ± 2.76</b>	<b>73.22 ± 1.74</b>	50.93 ± 0.82	61.71 ± 1.71	<b>66.58 ± 0.54</b>	<b>68.83 ± 6.35</b>

As shown in Table 15, the comparison between degree-based and random neighbor selection reveals distinct performance patterns across evaluation scenarios. Under clean conditions, SFT-neighbor demonstrates marginal advantages on most datasets. Interestingly, under text attack scenarios, SFT-rand exhibits superior robustness on several datasets, including Cora (78.60% vs 75.65%) and CiteSeer (47.08% vs 43.84%). However, under structural attacks, SFT-rand is more vulnerable, exhibiting larger performance drops. In conclusion, trade-offs also exist between the two variants.

#### I.1.2 LABEL INFORMATION IN PROMPTING

In (Wang et al., 2025b), incorporating label information in the prompt can yield better results for specific scenarios. We test the performance of such a prompt on clean datasets and against adversarial attacks. As shown in Table 16, incorporating explicit label information (SFT-neighbor-label) often leads to unstable performance, particularly on certain datasets like Instagram and Reddit, where severe degradation is observed across all scenarios. This lack of robustness and dataset-specific instability indicates potential overfitting or incompatibility with certain graph structures or data distributions. Consequently, label-enhanced prompting is not adopted due to its inconsistent performance.

## I.2 IMPACT OF DIFFERENT LLM BACKBONES

In the main paper, we use Mistral-7B as our LLM backbone as it shows the best performance in paper (Wu et al., 2025). In this subsection, we present the results of SFT with neighbor-aware prompts using different LLM backbones.

We use LLMs Mistral-7B (Jiang et al., 2023), Minstral-8B (Mistral AI Team, 2024), LLama3.1-8B (Dubey et al., 2024) and Qwen3-8B (Yang et al., 2025). The results are presented in Tables 17, 18, and 19.

The results demonstrate that:

Table 16: Comparison between standard degree-based selection (SFT-neighbor) and label-enhanced prompting (SFT-neighbor-label) across different attack scenarios. **Bolded values** indicate the better performance between the two methods for each dataset and scenario. Red values indicate significant performance degradation.

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History
<i>Clean Results on Inductive Split</i>							
SFT-neighbor	<b>88.25 ± 1.29</b>	76.17 ± 1.81	<b>95.08 ± 0.51</b>	<b>87.75 ± 0.46</b>	<b>69.24 ± 0.54</b>	<b>66.74 ± 1.06</b>	<b>86.81 ± 0.77</b>
SFT-neighbor-label	87.33 ± 2.50	<b>77.07 ± 0.72</b>	94.80 ± 0.44	<b>87.75 ± 0.66</b>	35.54 ± 3.54	14.08 ± 0.59	<b>86.81 ± 0.69</b>
<i>Structure Attack Results on Inductive Split</i>							
SFT-neighbor	82.35 ± 1.94	71.53 ± 1.86	<b>94.73 ± 0.41</b>	86.00 ± 1.40	<b>68.25 ± 0.20</b>	<b>62.29 ± 3.31</b>	85.98 ± 0.51
SFT-neighbor-label	<b>82.54 ± 1.67</b>	<b>73.35 ± 1.03</b>	94.41 ± 0.38	<b>86.28 ± 0.15</b>	45.84 ± 2.23	22.42 ± 2.01	<b>86.00 ± 0.78</b>
<i>Text Attack Results on Inductive Split</i>							
SFT-neighbor	<b>75.65 ± 1.33</b>	<b>43.84 ± 2.14</b>	72.27 ± 3.53	<b>51.53 ± 1.24</b>	<b>64.33 ± 1.46</b>	<b>65.52 ± 1.24</b>	<b>64.84 ± 0.98</b>
SFT-neighbor-label	64.51 ± 7.10	41.80 ± 0.77	<b>74.97 ± 4.68</b>	50.17 ± 0.58	34.64 ± 2.11	13.91 ± 0.59	64.58 ± 3.41

Table 17: Clean performance of SFT with neighbor-aware prompt in the inductive setting .

LLM	cora	CiteSeer	pubmed	wikics	history
Mistral-7B	<b>88.25 ± 1.29</b>	76.17 ± 1.81	95.08 ± 0.51	<b>87.75 ± 0.46</b>	<b>86.81 ± 0.77</b>
Minstral-8B	86.29 ± 0.91	76.70 ± 0.55	<b>95.50 ± 0.08</b>	87.28 ± 0.34	86.57 ± 0.85
Llama3-8B	86.10 ± 0.93	<b>77.43 ± 1.18</b>	95.38 ± 0.15	86.33 ± 0.15	86.68 ± 0.78
Qwen3-8B	85.55 ± 0.95	<b>77.07 ± 1.49</b>	95.18 ± 0.37	86.17 ± 0.91	86.08 ± 0.82

Table 18: SFT with neighbor prompt against structure attacks in the inductive setting.

LLM	cora	CiteSeer	pubmed	wikics	history
Mistral-7B	<b>82.35 ± 1.94</b>	71.53 ± 1.86	94.73 ± 0.41	<b>86.00 ± 1.40</b>	85.98 ± 0.51
Minstral-8B	79.64 ± 1.87	74.30 ± 0.98	<b>95.23 ± 0.07</b>	85.97 ± 0.91	86.06 ± 0.72
Llama3-8B	79.15 ± 1.12	<b>74.34 ± 0.48</b>	95.00 ± 0.14	84.98 ± 1.01	<b>86.08 ± 0.69</b>
Qwen3-8B	77.92 ± 1.68	72.94 ± 0.79	94.77 ± 0.49	84.73 ± 1.34	85.26 ± 0.70

Table 19: SFT with neighbor prompt against text attacks in the inductive setting.

LLM	cora	CiteSeer	pubmed	wikics	history
Mistral-7B	<b>75.65 ± 1.33</b>	43.84 ± 2.14	<b>72.27 ± 3.53</b>	<b>51.53 ± 1.24</b>	<b>64.84 ± 0.98</b>
Minstral-8B	68.76 ± 1.77	42.11 ± 1.02	71.02 ± 1.32	49.95 ± 0.44	55.15 ± 3.43
Llama3-8B	69.74 ± 3.81	43.37 ± 2.81	<b>72.19 ± 1.67</b>	49.58 ± 0.42	57.82 ± 3.37
Qwen3-8B	74.85 ± 2.33	<b>44.56 ± 2.69</b>	71.85 ± 0.56	49.44 ± 0.41	63.13 ± 3.19

- Mistral-7B consistently achieves the best performance across most datasets and settings, validating our choice of backbone in the main experiments
- Post-attack performance generally follows the same trends as clean performance, with models maintaining their relative advantages—for instance, Mistral-7B preserves its superiority on Cora across all attack scenarios
- Minor variations exist in specific contexts, such as on Cora and History datasets under textual attacks, where both Mistral-7B and Qwen3-8B demonstrate stronger robustness, though overall differences remain modest across backbones.

## J FULL EXPERIMENT RESULTS

### J.1 OVERVIEW AND CLARIFICATION OF RANK-BASED EVALUATION

The full results are provided in Section J.2 and Section J.3. In Section J.4, we include results against a smaller perturb ratio. We can see that compared to the main experiments, the attack effectiveness degrades.

In the main paper, we rely primarily on rank-based evaluation for the following reasons:

- Per-dataset plots are dense and difficult to read within the limited space of the main paper. Lines with similar performance tend to mix up.
- Averaging raw accuracies across datasets can be misleading due to scale differences and missing results on large-scale datasets, whereas rank provides a normalized, dataset-agnostic comparison.
- Raw accuracies are heavily influenced by clean accuracy, while rank mitigates such biases. Rank-based evaluation thus offers a clearer and more balanced summary of robustness across datasets.

To recover the rank-related results in the main paper, the procedure is as follows:

1. For each dataset, all methods with valid performance values are collected. A dash in the tables (–) indicates a method was not applicable due to OOM or scalability issues.
2. Methods are sorted within each dataset to determine their rank.
3. The final rank is the average of a method’s ranks across all datasets where it has values.

For example, a method with ranks (1, 2, 3) has a final rank of 2, while a method with ranks (1, 2, –) has a final rank of 1.5. Note that within each figure, only the methods included are considered during ranking.

### J.2 GNN AND RGNN RESULTS

#### J.2.1 CLEAN, INDUCTIVE

Results are in Tables 20, 22, 21, 23.

Table 20: Clean test accuracy under the inductive setting. (emb=BoW)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv	Avg Rank
GCN	85.98 ± 1.36	73.93 ± 1.94	86.84 ± 0.25	81.92 ± 0.29	63.05 ± 0.49	60.22 ± 0.82	81.97 ± 0.13	84.23 ± 0.17	87.75 ± 0.23	71.35 ± 0.00	5.40
GAT	86.22 ± 1.86	73.88 ± 0.88	86.72 ± 0.21	82.40 ± 0.62	64.67 ± 1.37	61.81 ± 1.15	82.07 ± 0.63	<b>84.75 ± 0.05</b>	88.09 ± 0.22	71.79 ± 0.00	3.80
APPNP	86.22 ± 1.40	71.94 ± 0.97	87.01 ± 0.21	80.52 ± 0.62	63.10 ± 1.59	57.27 ± 1.38	81.19 ± 0.43	82.99 ± 0.28	88.11 ± 0.16	71.27 ± 0.00	7.00
GPRGNN	83.95 ± 1.57	71.32 ± 1.63	<b>87.80 ± 0.02</b>	81.23 ± 0.32	64.34 ± 1.16	60.03 ± 0.72	78.59 ± 0.42	81.65 ± 0.66	85.02 ± 0.05	65.67 ± 0.00	8.60
RobustGCN	86.22 ± 2.01	73.77 ± 0.63	86.86 ± 0.36	82.88 ± 0.08	<b>66.74 ± 0.34</b>	58.06 ± 1.18	81.53 ± 0.57	84.03 ± 0.32	87.67 ± 0.17	68.20 ± 0.00	5.50
GCORN	83.33 ± 1.48	73.35 ± 0.59	86.73 ± 0.35	81.25 ± 0.22	64.98 ± 0.45	<b>65.01 ± 0.58</b>	80.92 ± 0.24	82.04 ± 0.08	84.53 ± 0.09	69.86 ± 0.00	7.10
NoisyGCN	86.35 ± 1.21	73.25 ± 1.36	86.73 ± 0.20	81.85 ± 0.31	63.12 ± 0.43	61.84 ± 0.31	<b>82.20 ± 0.25</b>	84.38 ± 0.16	87.75 ± 0.11	71.25 ± 0.00	5.00
GRAND	85.12 ± 1.76	<b>74.19 ± 1.41</b>	87.75 ± 0.45	81.49 ± 0.19	65.02 ± 0.48	62.50 ± 1.78	79.65 ± 0.62	81.71 ± 0.42	83.78 ± 0.14	68.23 ± 0.00	6.00
SoftmedianGDC	80.57 ± 1.90	73.35 ± 0.38	85.58 ± 0.08	78.48 ± 0.26	62.74 ± 1.15	61.33 ± 0.50	76.78 ± 0.46	–	–	–	10.57
EvenNet	84.81 ± 1.51	72.52 ± 0.70	87.63 ± 0.40	<b>83.01 ± 0.36</b>	65.24 ± 0.79	60.24 ± 0.11	81.01 ± 0.59	84.07 ± 0.03	87.86 ± 0.18	70.14 ± 0.00	5.40
ElasticGNN	<b>86.59 ± 1.97</b>	73.77 ± 0.07	87.60 ± 0.11	<b>83.11 ± 0.49</b>	<b>65.58 ± 1.04</b>	61.62 ± 0.43	81.71 ± 0.40	84.05 ± 0.08	<b>88.60 ± 0.12</b>	<b>72.24 ± 0.00</b>	2.80
GNNGuard	82.29 ± 1.23	69.85 ± 0.98	86.24 ± 0.32	75.05 ± 0.72	63.95 ± 0.64	54.37 ± 0.35	78.05 ± 0.38	77.81 ± 0.22	75.87 ± 0.35	70.49 ± 0.00	10.80
RUNG	81.73 ± 2.35	71.06 ± 0.27	85.72 ± 0.24	75.00 ± 0.25	64.40 ± 0.16	57.63 ± 0.60	–	–	–	–	11.17

Table 21: Clean test accuracy under the inductive setting. (emb=Mistral-7B)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv	Avg Rank
GCN	86.84 ± 1.99	75.44 ± 0.63	88.24 ± 0.63	83.43 ± 0.36	68.49 ± 0.30	65.39 ± 5.04	84.74 ± 0.44	85.64 ± 0.34	88.33 ± 0.16	74.18 ± 0.00	6.00
GAT	80.14 ± 1.03	71.58 ± 0.77	86.63 ± 0.62	66.28 ± 4.25	64.39 ± 0.27	61.14 ± 0.58	83.36 ± 0.61	82.83 ± 1.88	87.56 ± 0.48	72.91 ± 0.00	10.20
APPNP	84.69 ± 1.20	75.91 ± 0.82	92.02 ± 0.22	85.79 ± 0.46	69.02 ± 0.18	63.02 ± 0.51	84.88 ± 0.56	85.53 ± 0.49	88.51 ± 0.40	<b>76.22 ± 0.00</b>	4.40
GPRGNN	85.85 ± 0.98	<b>77.12 ± 0.38</b>	<b>92.44 ± 0.22</b>	86.50 ± 0.65	67.75 ± 0.95	65.56 ± 0.51	84.82 ± 0.38	84.75 ± 0.62	87.84 ± 0.16	76.11 ± 0.00	4.20
RobustGCN	58.61 ± 8.78	46.03 ± 3.30	45.01 ± 7.14	27.01 ± 5.97	64.26 ± 0.53	52.75 ± 0.27	56.16 ± 0.43	42.14 ± 0.11	25.89 ± 0.18	5.86 ± 0.00	11.60
GCORN	86.16 ± 1.98	76.33 ± 0.78	90.50 ± 0.39	85.69 ± 0.39	66.18 ± 0.35	<b>69.28 ± 0.88</b>	<b>85.26 ± 0.31</b>	<b>86.36 ± 0.27</b>	87.67 ± 0.25	75.52 ± 0.00	4.50
NoisyGCN	87.95 ± 1.30	75.18 ± 0.27	88.23 ± 0.03	84.00 ± 0.29	68.83 ± 0.66	65.38 ± 3.55	84.74 ± 0.43	85.95 ± 0.35	88.44 ± 0.39	74.29 ± 0.00	5.30
GRAND	86.29 ± 1.45	76.49 ± 0.64	90.80 ± 0.33	85.32 ± 1.05	68.25 ± 0.10	63.55 ± 0.88	84.57 ± 0.57	77.99 ± 3.35	79.59 ± 1.60	–	6.56
SoftmedianGDC	82.35 ± 1.60	73.04 ± 0.26	91.70 ± 0.18	81.48 ± 3.09	–	60.79 ± 0.42	84.50 ± 0.61	–	–	–	9.33
EvenNet	85.79 ± 1.04	75.60 ± 0.70	90.77 ± 0.07	<b>87.13 ± 0.53</b>	<b>69.34 ± 0.24</b>	64.91 ± 0.30	84.90 ± 0.53	85.00 ± 0.29	88.34 ± 0.62	74.12 ± 0.00	4.50
ElasticGNN	<b>88.25 ± 1.30</b>	76.70 ± 0.91	89.33 ± 0.46	86.62 ± 0.34	68.74 ± 0.47	49.95 ± 0.31	84.90 ± 0.49	85.60 ± 0.46	<b>88.96 ± 0.44</b>	76.09 ± 0.00	4.20
GNNGuard	85.49 ± 0.74	74.35 ± 0.52	90.20 ± 0.23	84.25 ± 1.71	69.12 ± 0.21	59.48 ± 7.12	–	–	–	–	7.50
RUNG	82.90 ± 1.81	73.67 ± 0.38	82.34 ± 0.06	83.53 ± 0.65	68.50 ± 0.08	63.60 ± 0.49	–	–	–	–	7.17

Table 22: Clean test accuracy under the inductive setting. (emb=MiniLM)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv	Avg Rank
GCN	86.35 $\pm$ 2.07	75.76 $\pm$ 0.98	89.28 $\pm$ 0.51	83.81 $\pm$ 0.13	65.06 $\pm$ 0.68	66.98 $\pm$ 0.59	84.57 $\pm$ 0.36	86.04 $\pm$ 0.25	88.93 $\pm$ 0.11	73.86 $\pm$ 0.00	5.60
GAT	87.33 $\pm$ 1.58	74.87 $\pm$ 0.85	87.53 $\pm$ 0.08	83.47 $\pm$ 0.21	67.11 $\pm$ 0.28	65.00 $\pm$ 0.85	83.84 $\pm$ 0.53	86.14 $\pm$ 0.41	89.10 $\pm$ 0.17	73.31 $\pm$ 0.00	6.20
APPNP	84.93 $\pm$ 1.92	75.76 $\pm$ 0.87	89.22 $\pm$ 0.32	83.44 $\pm$ 0.56	66.05 $\pm$ 0.06	58.83 $\pm$ 0.60	84.73 $\pm$ 0.59	85.28 $\pm$ 0.16	89.29 $\pm$ 0.10	75.71 $\pm$ 0.00	5.90
GPRGNN	86.41 $\pm$ 1.67	75.60 $\pm$ 0.53	89.96 $\pm$ 0.26	82.41 $\pm$ 0.64	63.10 $\pm$ 0.88	59.08 $\pm$ 0.08	83.12 $\pm$ 0.50	83.77 $\pm$ 0.25	87.75 $\pm$ 0.15	71.97 $\pm$ 0.00	8.60
RobustGCN	<b>87.95 <math>\pm</math> 1.03</b>	76.18 $\pm$ 1.05	88.07 $\pm$ 0.19	84.12 $\pm$ 0.05	<b>67.28 <math>\pm</math> 0.16</b>	58.65 $\pm$ 0.21	84.86 $\pm$ 0.38	85.97 $\pm$ 0.21	88.83 $\pm$ 0.15	74.09 $\pm$ 0.00	5.00
GCORN	86.72 $\pm$ 2.04	76.07 $\pm$ 1.15	88.80 $\pm$ 0.03	83.51 $\pm$ 0.30	64.62 $\pm$ 0.61	67.08 $\pm$ 0.51	84.72 $\pm$ 0.47	84.37 $\pm$ 0.21	85.84 $\pm$ 0.08	74.35 $\pm$ 0.00	5.90
NoisyGCN	85.92 $\pm$ 1.77	76.07 $\pm$ 0.48	89.22 $\pm$ 0.18	84.07 $\pm$ 0.08	65.14 $\pm$ 0.54	<b>67.19 <math>\pm</math> 0.52</b>	84.54 $\pm$ 0.38	<b>86.15 <math>\pm</math> 0.32</b>	89.08 $\pm$ 0.20	73.57 $\pm$ 0.00	5.20
GRAND	86.41 $\pm$ 1.28	<b>77.48 <math>\pm</math> 1.74</b>	88.15 $\pm$ 0.14	83.06 $\pm$ 0.08	65.46 $\pm$ 0.31	64.60 $\pm$ 0.49	83.03 $\pm$ 0.52	82.40 $\pm$ 0.46	83.49 $\pm$ 0.08	72.22 $\pm$ 0.00	7.90
SoftmedianGDC	86.47 $\pm$ 1.61	76.65 $\pm$ 0.34	<b>90.31 <math>\pm</math> 0.27</b>	82.22 $\pm$ 0.49	61.60 $\pm$ 0.57	63.43 $\pm$ 0.13	83.64 $\pm$ 0.53	—	—	—	7.00
EvenNet	86.78 $\pm$ 1.26	75.76 $\pm$ 2.25	88.97 $\pm$ 0.07	<b>84.76 <math>\pm</math> 0.22</b>	65.95 $\pm$ 0.81	61.46 $\pm$ 0.20	84.63 $\pm$ 0.46	85.56 $\pm$ 0.22	89.15 $\pm$ 0.14	73.78 $\pm$ 0.00	5.10
ElasticGNN	86.53 $\pm$ 1.98	77.06 $\pm$ 1.26	88.67 $\pm$ 0.20	<b>84.54 <math>\pm</math> 0.27</b>	66.68 $\pm$ 0.34	63.53 $\pm$ 0.41	<b>84.98 <math>\pm</math> 0.54</b>	85.70 $\pm$ 0.25	<b>89.66 <math>\pm</math> 0.09</b>	<b>75.82 <math>\pm</math> 0.00</b>	3.60
GNNGuard	82.84 $\pm$ 1.14	75.08 $\pm$ 0.46	88.88 $\pm$ 0.06	80.72 $\pm$ 0.61	60.95 $\pm$ 1.04	56.83 $\pm$ 0.60	82.92 $\pm$ 0.15	74.70 $\pm$ 0.25	79.01 $\pm$ 0.08	71.25 $\pm$ 0.00	11.60
RUNG	84.75 $\pm$ 1.65	73.46 $\pm$ 0.87	89.78 $\pm$ 0.14	81.48 $\pm$ 0.38	63.51 $\pm$ 0.38	58.72 $\pm$ 0.51	—	—	—	—	10.17

Table 23: Clean test accuracy under the inductive setting. (emb=RoBERTa)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv	Avg Rank
GCN	87.76 $\pm$ 1.51	75.60 $\pm$ 1.09	88.76 $\pm$ 0.25	84.86 $\pm$ 0.61	66.75 $\pm$ 0.29	67.76 $\pm$ 0.74	85.00 $\pm$ 0.31	86.46 $\pm$ 0.26	89.24 $\pm$ 0.15	73.29 $\pm$ 0.00	6.20
GAT	87.70 $\pm$ 1.30	<b>76.59 <math>\pm</math> 0.45</b>	88.14 $\pm$ 0.48	84.42 $\pm$ 0.12	<b>67.80 <math>\pm</math> 0.23</b>	64.64 $\pm$ 0.22	84.66 $\pm$ 0.67	86.30 $\pm$ 0.17	<b>89.70 <math>\pm</math> 0.28</b>	74.09 $\pm$ 0.00	5.70
APPNP	84.75 $\pm$ 1.61	75.76 $\pm$ 0.63	<b>91.26 <math>\pm</math> 0.22</b>	85.03 $\pm$ 0.63	67.21 $\pm$ 0.13	59.99 $\pm$ 0.45	85.79 $\pm$ 0.57	85.70 $\pm$ 0.20	88.99 $\pm$ 0.14	76.00 $\pm$ 0.00	5.80
GPRGNN	86.29 $\pm$ 1.61	74.19 $\pm$ 0.45	91.05 $\pm$ 0.16	84.38 $\pm$ 1.00	64.26 $\pm$ 0.58	58.35 $\pm$ 0.67	84.68 $\pm$ 0.45	84.76 $\pm$ 0.23	88.23 $\pm$ 0.17	71.83 $\pm$ 0.00	9.70
RobustGCN	87.70 $\pm$ 1.28	74.82 $\pm$ 0.59	87.92 $\pm$ 0.12	84.37 $\pm$ 0.18	<b>67.97 <math>\pm</math> 0.08</b>	63.09 $\pm$ 0.87	84.43 $\pm$ 0.43	84.79 $\pm$ 0.39	86.78 $\pm$ 0.14	73.73 $\pm$ 0.00	8.50
GCORN	87.52 $\pm$ 1.15	76.28 $\pm$ 0.15	89.83 $\pm$ 0.16	85.48 $\pm$ 0.27	65.17 $\pm$ 0.52	<b>67.85 <math>\pm</math> 0.48</b>	85.28 $\pm$ 0.49	85.64 $\pm$ 0.05	87.51 $\pm$ 0.06	73.69 $\pm$ 0.00	6.30
NoisyGCN	88.01 $\pm$ 1.69	75.86 $\pm$ 0.51	88.75 $\pm$ 0.32	84.35 $\pm$ 0.61	67.12 $\pm$ 0.32	67.75 $\pm$ 0.78	84.98 $\pm$ 0.08	86.37 $\pm$ 0.22	89.23 $\pm$ 0.25	73.35 $\pm$ 0.00	6.50
GRAND	<b>88.75 <math>\pm</math> 0.94</b>	<b>77.85 <math>\pm</math> 1.29</b>	90.47 $\pm$ 0.24	<b>85.90 <math>\pm</math> 0.31</b>	67.64 $\pm$ 0.26	65.83 $\pm$ 0.49	<b>86.00 <math>\pm</math> 0.36</b>	85.54 $\pm$ 0.22	88.13 $\pm$ 0.02	75.85 $\pm$ 0.00	3.80
SoftmedianGDC	85.36 $\pm$ 2.13	75.71 $\pm$ 0.90	90.76 $\pm$ 0.22	83.53 $\pm$ 0.99	65.53 $\pm$ 0.50	62.91 $\pm$ 0.19	84.78 $\pm$ 0.47	—	—	—	8.71
EvenNet	86.35 $\pm$ 1.18	75.86 $\pm$ 0.71	90.74 $\pm$ 0.15	<b>85.95 <math>\pm</math> 0.63</b>	67.68 $\pm$ 0.65	61.65 $\pm$ 0.79	85.63 $\pm$ 0.40	<b>86.55 <math>\pm</math> 0.16</b>	89.51 $\pm$ 0.21	74.75 $\pm$ 0.00	4.20
ElasticGNN	<b>88.31 <math>\pm</math> 1.51</b>	75.76 $\pm$ 0.52	89.89 $\pm$ 0.20	85.48 $\pm$ 0.38	67.64 $\pm$ 0.80	64.06 $\pm$ 0.87	<b>85.81 <math>\pm</math> 0.57</b>	86.19 $\pm$ 0.02	<b>90.07 <math>\pm</math> 0.08</b>	<b>76.22 <math>\pm</math> 0.00</b>	3.80
GNNGuard	83.64 $\pm$ 1.94	74.03 $\pm$ 0.53	90.21 $\pm$ 0.41	83.03 $\pm$ 0.53	64.09 $\pm$ 1.23	58.58 $\pm$ 0.33	84.65 $\pm$ 0.49	83.07 $\pm$ 0.08	82.38 $\pm$ 0.08	71.16 $\pm$ 0.00	12.10
RUNG	84.93 $\pm$ 1.46	73.30 $\pm$ 1.19	90.72 $\pm$ 0.25	84.37 $\pm$ 0.44	66.43 $\pm$ 0.27	58.80 $\pm$ 0.97	—	—	—	—	10.17
Guardual	84.26 $\pm$ 1.71	74.71 $\pm$ 1.22	89.83 $\pm$ 0.29	84.54 $\pm$ 0.34	65.53 $\pm$ 0.32	62.24 $\pm$ 0.22	85.01 $\pm$ 0.28	79.78 $\pm$ 0.17	77.39 $\pm$ 0.31	72.17 $\pm$ 0.00	9.90

### J.2.2 CLEAN, TRANSDUCTIVE

Results are in Tables 24, 25, 26.

Table 24: Clean test accuracy under the transductive setting. (ptb\_rate=0.3, atk\_emb=BoW, def\_emb=BoW)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv	Avg Rank
GCN	83.19 $\pm$ 0.75	71.32 $\pm$ 0.83	85.45 $\pm$ 0.24	80.01 $\pm$ 0.29	65.97 $\pm$ 0.09	61.84 $\pm$ 0.49	80.14 $\pm$ 0.25	—	—	—	6.86
GAT	82.99 $\pm$ 1.03	72.30 $\pm$ 0.56	85.38 $\pm$ 0.21	79.76 $\pm$ 0.35	65.41 $\pm$ 0.06	60.73 $\pm$ 0.30	80.67 $\pm$ 0.12	—	—	—	7.86
APPNP	83.69 $\pm$ 0.30	72.25 $\pm$ 1.10	<b>87.34 <math>\pm</math> 0.08</b>	80.49 $\pm$ 0.33	65.78 $\pm$ 0.10	60.78 $\pm$ 0.66	80.84 $\pm$ 0.06	—	—	—	4.14
GPRGNN	83.06 $\pm$ 0.73	<b>72.56 <math>\pm</math> 0.65</b>	86.48 $\pm$ 0.24	79.80 $\pm$ 0.14	<b>66.12 <math>\pm</math> 0.18</b>	60.37 $\pm$ 1.08	80.49 $\pm$ 0.05	—	—	—	5.43
RobustGCN	<b>83.77 <math>\pm</math> 0.80</b>	71.64 $\pm$ 0.66	86.45 $\pm$ 0.30	<b>81.15 <math>\pm</math> 0.14</b>	65.94 $\pm$ 0.31	59.29 $\pm$ 0.82	<b>80.95 <math>\pm</math> 0.15</b>	—	—	—	5.43
GCORN	77.40 $\pm$ 0.60	69.77 $\pm$ 0.58	84.71 $\pm$ 0.32	78.87 $\pm$ 0.33	65.43 $\pm$ 0.45	<b>66.34 <math>\pm</math> 0.08</b>	78.36 $\pm$ 0.02	—	—	—	11.57
NoisyGCN	83.19 $\pm$ 0.12	72.36 $\pm$ 0.94	85.75 $\pm$ 0.42	79.74 $\pm$ 0.35	65.77 $\pm$ 0.15	63.83 $\pm$ 0.13	80.34 $\pm$ 0.32	—	—	—	6.00
GRAND	<b>83.97 <math>\pm</math> 0.92</b>	71.71 $\pm$ 0.14	87.16 $\pm$ 0.30	80.71 $\pm$ 0.38	65.94 $\pm$ 0.27	61.58 $\pm$ 1.33	80.35 $\pm$ 0.27	—	—	—	4.43
SoftmedianGDC	82.77 $\pm$ 1.35	71.99 $\pm$ 0.28	85.15 $\pm$ 0.07	74.33 $\pm$ 0.49	<b>65.98 <math>\pm</math> 0.40</b>	59.67 $\pm$ 0.52	78.04 $\pm$ 0.08	—	—	—	10.57
EvenNet	82.88 $\pm$ 0.56	71.70 $\pm$ 0.70	86.50 $\pm$ 0.24	80.71 $\pm$ 0.31	<b>65.77 <math>\pm</math> 0.72</b>	<b>64.61 <math>\pm</math> 0.68</b>	80.48 $\pm$ 0.07	—	—	—	5.57
ElasticGNN	83.19 $\pm$ 0.17	<b>73.14 <math>\pm</math> 0.40</b>	87.00 $\pm$ 0.17	80.18 $\pm$ 0.27	65.41 $\pm$ 0.37	60.58 $\pm$ 0.47	80.67 $\pm$ 0.07	—	—	—	5.43
GNNGuard	80.31 $\pm$ 0.37	71.20 $\pm$ 0.28	86.13 $\pm$ 0.26	77.08 $\pm$ 0.28	64.26 $\pm$ 0.79	58.79 $\pm$ 0.43	79.54 $\pm$ 0.18	—	—	—	12.71
RUNG	81.91 $\pm$ 0.59	72.20 $\pm$ 0.80	85.57 $\pm$ 0.30	78.75 $\pm$ 0.47	65.55 $\pm$ 0.23	60.24 $\pm$ 0.11	—	—	—	—	10.17
Cosine-GCN	79.48 $\pm$ 1.38	69.70 $\pm$ 0.92	84.03 $\pm$ 0.26	77.61 $\pm$ 0.47	64.78 $\pm$ 0.50	59.07 $\pm$ 0.36	78.26 $\pm$ 0.21	—	—	—	14.57
Jaccard-GCN	80.60 $\pm$ 1.25	69.98 $\pm$ 0.82	84.12 $\pm$ 0.34	79.11 $\pm$ 0.29	65.66 $\pm$ 0.13	61.77 $\pm$ 0.26	78.44 $\pm$ 0.19	—	—	—	10.86
Stable	79.97 $\pm$ 0.45	68.29 $\pm$ 0.67	84.00 $\pm$ 0.43	78.94 $\pm$ 0.55	64.66 $\pm$ 0.51	63.32 $\pm$ 0.31	—	—	—	—	12.83
ProGNN	77.79 $\pm$ 0.19	70.13 $\pm$ 0.93	—	—	—	—	—	—	—	—	14.50

### J.2.3 STRUCTURE ATTACK, INDUCTIVE

Results are in Tables 27, 29, 28, 30.

### J.2.4 STRUCTURE ATTACK, TRANSDUCTIVE

Results are in Tables 31, 32, 33.

### J.2.5 TEXT ATTACK, INDUCTIVE

Results are in Tables

Table 25: Clean test accuracy under the transductive setting. (ptb\_rate=0.3, atk\_emb=BoW, def\_emb=MiniLM)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Avg Rank
GCN	84.34 $\pm$ 0.34	74.32 $\pm$ 0.88	86.78 $\pm$ 0.48	81.31 $\pm$ 0.05	65.67 $\pm$ 0.33	65.81 $\pm$ 0.63	82.94 $\pm$ 0.11	8.43
GAT	83.97 $\pm$ 1.30	73.86 $\pm$ 0.61	85.94 $\pm$ 0.35	81.85 $\pm$ 0.31	65.56 $\pm$ 0.07	63.31 $\pm$ 0.86	82.65 $\pm$ 0.01	10.29
APPNP	84.33 $\pm$ 0.79	74.43 $\pm$ 0.36	<b>88.36 <math>\pm</math> 0.26</b>	81.59 $\pm$ 0.28	65.68 $\pm$ 0.44	62.54 $\pm$ 0.09	83.44 $\pm$ 0.10	5.14
GPRGNN	84.31 $\pm$ 0.17	73.98 $\pm$ 0.91	88.04 $\pm$ 0.21	81.45 $\pm$ 0.23	65.21 $\pm$ 0.30	62.34 $\pm$ 0.25	82.61 $\pm$ 0.09	10.00
RobustGCN	84.05 $\pm$ 0.51	75.10 $\pm$ 0.65	86.92 $\pm$ 0.12	81.82 $\pm$ 0.30	<b>66.14 <math>\pm</math> 0.41</b>	64.05 $\pm$ 0.29	83.18 $\pm$ 0.06	5.86
GCORN	82.77 $\pm$ 0.80	74.94 $\pm$ 0.65	86.99 $\pm$ 0.14	81.07 $\pm$ 0.35	65.75 $\pm$ 0.15	<b>67.37 <math>\pm</math> 0.29</b>	83.12 $\pm$ 0.02	7.86
NoisyGCN	84.33 $\pm$ 0.44	74.11 $\pm$ 0.59	87.15 $\pm$ 0.23	81.42 $\pm$ 0.07	65.46 $\pm$ 0.01	<b>67.52 <math>\pm</math> 0.25</b>	82.84 $\pm$ 0.12	9.00
GRAND	84.33 $\pm$ 0.53	<b>75.82 <math>\pm</math> 0.49</b>	87.38 $\pm$ 0.16	81.69 $\pm$ 0.22	65.51 $\pm$ 0.24	65.02 $\pm$ 0.83	83.10 $\pm$ 0.18	5.86
SoftmedianGDC	84.42 $\pm$ 0.25	75.47 $\pm$ 0.45	88.16 $\pm$ 0.30	81.04 $\pm$ 0.38	65.88 $\pm$ 0.28	63.72 $\pm$ 0.29	<b>83.54 <math>\pm</math> 0.10</b>	4.57
EvenNet	84.39 $\pm$ 0.60	74.75 $\pm$ 0.99	87.45 $\pm$ 0.23	82.11 $\pm$ 0.59	65.60 $\pm$ 0.23	62.01 $\pm$ 0.42	83.28 $\pm$ 0.07	5.86
ElasticGNN	<b>84.63 <math>\pm</math> 0.15</b>	74.43 $\pm$ 0.36	87.69 $\pm$ 0.05	81.47 $\pm$ 0.23	65.77 $\pm$ 0.31	62.54 $\pm$ 0.14	83.24 $\pm$ 0.05	5.43
GNNGuard	82.71 $\pm$ 0.29	74.36 $\pm$ 0.58	87.37 $\pm$ 0.10	81.55 $\pm$ 0.01	64.79 $\pm$ 0.67	59.29 $\pm$ 0.57	82.75 $\pm$ 0.26	11.57
RUNG	82.19 $\pm$ 0.26	73.68 $\pm$ 0.20	87.76 $\pm$ 0.26	81.37 $\pm$ 0.01	65.02 $\pm$ 0.53	61.51 $\pm$ 0.11	–	12.67
Cosine-GCN	82.02 $\pm$ 0.81	74.19 $\pm$ 0.34	86.30 $\pm$ 0.22	81.36 $\pm$ 0.09	65.45 $\pm$ 0.38	59.37 $\pm$ 0.08	82.50 $\pm$ 0.08	13.43
Jaccard-GCN	83.42 $\pm$ 0.19	73.85 $\pm$ 0.73	86.30 $\pm$ 0.26	80.94 $\pm$ 0.16	65.67 $\pm$ 0.41	66.10 $\pm$ 0.33	82.30 $\pm$ 0.06	11.57
Stable	84.09 $\pm$ 0.19	73.68 $\pm$ 1.06	86.84 $\pm$ 0.23	<b>82.53 <math>\pm</math> 0.28</b>	65.33 $\pm$ 0.54	66.21 $\pm$ 0.24	–	8.83
ProGNN	83.20 $\pm$ 0.51	75.34 $\pm$ 0.72	–	–	–	–	–	8.00

Table 26: Clean test accuracy under the transductive setting. (ptb\_rate=0.3, atk\_emb=BoW, def\_emb=RoBERTa)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Avg Rank
GCN	84.03 $\pm$ 0.51	73.98 $\pm$ 0.70	87.39 $\pm$ 0.11	81.85 $\pm$ 0.46	66.68 $\pm$ 0.20	63.41 $\pm$ 0.12	83.55 $\pm$ 0.20	10.57
GAT	84.25 $\pm$ 0.48	74.19 $\pm$ 0.40	86.20 $\pm$ 0.20	82.05 $\pm$ 0.25	66.45 $\pm$ 0.24	62.91 $\pm$ 0.19	83.33 $\pm$ 0.21	12.29
APPNP	85.03 $\pm$ 0.24	74.74 $\pm$ 0.22	<b>89.13 <math>\pm</math> 0.04</b>	82.51 $\pm$ 0.55	<b>67.48 <math>\pm</math> 0.11</b>	63.80 $\pm$ 0.15	84.43 $\pm$ 0.20	3.43
GPRGNN	85.05 $\pm$ 0.36	74.42 $\pm$ 0.50	89.06 $\pm$ 0.28	82.17 $\pm$ 0.44	67.26 $\pm$ 0.13	63.74 $\pm$ 0.12	84.05 $\pm$ 0.21	4.86
RobustGCN	83.54 $\pm$ 0.91	74.55 $\pm$ 1.12	87.27 $\pm$ 0.07	81.83 $\pm$ 0.19	66.62 $\pm$ 0.25	62.73 $\pm$ 0.74	83.20 $\pm$ 0.09	12.43
GCORN	84.34 $\pm$ 0.61	74.67 $\pm$ 1.32	87.44 $\pm$ 0.05	82.17 $\pm$ 0.15	66.46 $\pm$ 0.30	<b>67.60 <math>\pm</math> 0.18</b>	83.81 $\pm$ 0.10	7.86
NoisyGCN	84.59 $\pm$ 0.75	73.69 $\pm$ 0.77	87.29 $\pm$ 0.23	82.12 $\pm$ 0.18	66.66 $\pm$ 0.61	63.87 $\pm$ 0.14	83.55 $\pm$ 0.16	9.71
GRAND	<b>85.16 <math>\pm</math> 0.61</b>	<b>76.06 <math>\pm</math> 0.62</b>	89.11 $\pm$ 0.08	<b>84.16 <math>\pm</math> 0.31</b>	66.94 $\pm$ 0.47	<b>66.34 <math>\pm</math> 0.61</b>	<b>84.93 <math>\pm</math> 0.09</b>	2.00
SoftmedianGDC	83.56 $\pm$ 0.61	74.11 $\pm$ 0.43	88.48 $\pm$ 0.21	80.22 $\pm$ 0.67	66.93 $\pm$ 0.21	63.10 $\pm$ 0.40	84.30 $\pm$ 0.13	9.43
EvenNet	84.06 $\pm$ 1.02	74.94 $\pm$ 0.60	88.99 $\pm$ 0.14	<b>83.15 <math>\pm</math> 0.53</b>	67.05 $\pm$ 0.16	63.23 $\pm$ 0.51	83.90 $\pm$ 0.14	5.29
ElasticGNN	84.88 $\pm$ 0.19	74.41 $\pm$ 1.05	88.28 $\pm$ 0.17	82.65 $\pm$ 0.23	66.87 $\pm$ 0.20	63.17 $\pm$ 0.13	83.99 $\pm$ 0.18	6.57
GNNGuard	83.22 $\pm$ 0.55	73.81 $\pm$ 0.50	87.98 $\pm$ 0.29	82.47 $\pm$ 0.19	67.23 $\pm$ 0.31	59.39 $\pm$ 0.39	83.87 $\pm$ 0.15	10.14
RUNG	83.42 $\pm$ 0.64	75.35 $\pm$ 0.88	88.86 $\pm$ 0.26	82.64 $\pm$ 0.33	66.82 $\pm$ 0.27	62.85 $\pm$ 0.17	–	8.17
Cosine-GCN	83.25 $\pm$ 1.08	73.43 $\pm$ 0.61	87.05 $\pm$ 0.16	82.60 $\pm$ 0.63	66.96 $\pm$ 0.21	59.37 $\pm$ 0.49	83.75 $\pm$ 0.20	11.43
Jaccard-GCN	83.54 $\pm$ 0.77	72.98 $\pm$ 0.42	87.04 $\pm$ 0.15	81.47 $\pm$ 0.42	66.70 $\pm$ 0.50	63.41 $\pm$ 0.12	83.18 $\pm$ 0.05	12.86
Stable	84.40 $\pm$ 0.80	72.70 $\pm$ 1.30	86.66 $\pm$ 0.27	82.65 $\pm$ 0.36	66.75 $\pm$ 0.26	65.33 $\pm$ 0.58	–	9.00
ProGNN	83.57 $\pm$ 0.79	74.91 $\pm$ 0.55	–	–	–	–	–	7.50

Table 27: Accuracy under the inductive/evasion setting against structural attack. (ptb\_rate=0.2, atk\_emb=BoW, def\_emb=BoW)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv	Avg Rank
GCN	66.11 $\pm$ 2.48	54.86 $\pm$ 1.00	80.12 $\pm$ 0.71	20.39 $\pm$ 0.56	7.11 $\pm$ 0.81	50.47 $\pm$ 3.28	41.70 $\pm$ 0.39	38.88 $\pm$ 2.42	35.95 $\pm$ 0.33	32.86 $\pm$ 0.00	11.60
GAT	67.53 $\pm$ 2.03	57.31 $\pm$ 1.07	80.18 $\pm$ 0.57	39.47 $\pm$ 5.36	45.03 $\pm$ 5.36	53.83 $\pm$ 3.18	49.44 $\pm$ 1.63	51.95 $\pm$ 1.37	44.08 $\pm$ 0.03	39.62 $\pm$ 0.00	8.10
APPNP	69.13 $\pm$ 2.82	64.37 $\pm$ 0.99	83.75 $\pm$ 0.14	57.37 $\pm$ 1.22	55.61 $\pm$ 2.66	52.40 $\pm$ 2.04	59.33 $\pm$ 0.76	55.21 $\pm$ 2.26	41.06 $\pm$ 0.19	43.12 $\pm$ 0.00	5.40
GPRGNN	71.83 $\pm$ 1.22	62.49 $\pm$ 0.60	82.53 $\pm$ 0.25	57.64 $\pm$ 0.86	60.96 $\pm$ 1.02	<b>64.21 <math>\pm</math> 1.65</b>	64.79 $\pm$ 0.97	<b>59.31 <math>\pm</math> 0.10</b>	<b>50.72 <math>\pm</math> 0.45</b>	<b>50.18 <math>\pm</math> 0.00</b>	2.90
RobustGCN	69.99 $\pm$ 2.80	58.25 $\pm$ 1.48	80.33 $\pm$ 0.61	31.14 $\pm$ 0.82	60.89 $\pm$ 0.47	52.74 $\pm$ 1.37	51.27 $\pm$ 0.75	48.72 $\pm$ 0.90	40.05 $\pm$ 0.94	41.67 $\pm$ 0.00	7.90
GCORN	68.57 $\pm$ 2.52	58.52 $\pm$ 0.97	81.69 $\pm$ 0.50	34.16 $\pm$ 0.21	32.63 $\pm$ 2.84	59.81 $\pm$ 1.73	41.51 $\pm$ 0.77	39.00 $\pm$ 0.67	37.12 $\pm$ 0.15	31.39 $\pm$ 0.00	8.90
NoisyGCN	66.24 $\pm$ 2.87	54.49 $\pm$ 0.73	80.27 $\pm$ 0.37	20.38 $\pm$ 0.57	6.08 $\pm$ 0.55	55.26 $\pm$ 1.17	41.41 $\pm$ 0.76	39.05 $\pm$ 1.28	36.61 $\pm$ 0.33	34.48 $\pm$ 0.00	10.90
GRAND	71.89 $\pm$ 3.25	64.00 $\pm$ 0.93	82.45 $\pm$ 0.08	48.57 $\pm$ 1.07	<b>64.15 <math>\pm</math> 0.37</b>	58.74 $\pm$ 3.02	47.45 $\pm$ 0.66	49.11 $\pm$ 0.74	45.02 $\pm$ 0.16	<b>49.90 <math>\pm</math> 0.00</b>	4.60
SoftmedianGDC	74.11 $\pm$ 2.93	64.32 $\pm$ 1.07	82.35 $\pm$ 0.25	72.66 $\pm$ 0.24	55.28 $\pm$ 1.11	<b>61.88 <math>\pm</math> 0.92</b>	<b>69.80 <math>\pm</math> 0.49</b>	–	–	–	3.86
EvenNet	70.42 $\pm$ 2.27	61.86 $\pm$ 1.03	82.47 $\pm$ 0.33	32.88 $\pm$ 1.56	62.38 $\pm$ 1.38	60.10 $\pm$ 1.03	61.94 $\pm$ 0.41	54.04 $\pm$ 0.65	<b>46.07 <math>\pm</math> 0.72</b>	38.01 $\pm$ 0.00	5.10
ElasticGNN	68.51 $\pm$ 2.73	64.53 $\pm$ 0.00	81.07 $\pm$ 0.64	37.33 $\pm$ 0.24	39.99 $\pm$ 6.61	45.74 $\pm$ 0.54	60.02 $\pm$ 0.80	55.96 $\pm$ 0.40	40.43 $\pm$ 0.38	43.43 $\pm$ 0.00	7.60
GNNGuard	67.16 $\pm$ 2.91	<b>68.03 <math>\pm</math> 0.90</b>	84.56 $\pm$ 0.14	<b>75.02 <math>\pm</math> 0.71</b>	63.57 $\pm$ 0.58	53.91 $\pm$ 0.30	53.70 $\pm$ 0.28	46.16 $\pm$ 0.33	44.98 $\pm$ 0.28	39.25 $\pm$ 0.00	5.00
RUNG	<b>78.04 <math>\pm</math> 2.22</b>	67.97 $\pm$ 1.37	<b>85.70 <math>\pm</math> 0.25</b>	75.00 $\pm$ 0.25	62.42 $\pm$ 0.08	55.41 $\pm$ 0.58	–	–	–	–	2.50

Table 28: Accuracy under the inductive/evasion setting against structural attack. (ptb\_rate=0.2, atk\_emb=BoW, def\_emb=Mistral-7B)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv	Avg Rank
GCN	73.06 $\pm$ 2.89	63.06 $\pm$ 1.54	82.33 $\pm$ 1.07	34.86 $\pm$ 1.97	62.23 $\pm$ 1.33	62.90 $\pm$ 9.60	64.32 $\pm$ 1.25	55.53 $\pm$ 0.58	51.20 $\pm$ 0.30	47.70 $\pm$ 0.00	8.10
GAT	67.59 $\pm$ 1.23	61.39 $\pm$ 1.56	81.65 $\pm$ 1.16	38.33 $\pm$ 3.51	64.23 $\pm$ 0.56	57.27 $\pm$ 0.44	65.34 $\pm$ 1.52	54.54 $\pm$ 0.27	51.71 $\pm$ 1.72	49.98 $\pm$ 0.00	8.60
APPNP	78.54 $\pm$ 0.71	69.70 $\pm$ 1.09	90.35 $\pm$ 0.26	79.27 $\pm$ 1.00	68.52 $\pm$ 0.35	60.18 $\pm$ 0.63	76.83 $\pm$ 0.83	64.66 $\pm$ 1.12	50.41 $\pm$ 1.33	<b>59.74 <math>\pm</math> 0.00</b>	3.50</

Table 29: Accuracy under the inductive/evasion setting against structural attack. (ptb\_rate=0.2, atk\_emb=BoW, def\_emb=MiniLM)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv	Avg Rank
GCN	72.39 $\pm$ 2.45	61.34 $\pm$ 1.31	83.87 $\pm$ 0.87	30.44 $\pm$ 1.40	57.58 $\pm$ 0.72	65.79 $\pm$ 1.80	64.83 $\pm$ 0.83	58.03 $\pm$ 0.53	55.90 $\pm$ 0.31	41.35 $\pm$ 0.00	9.60
GAT	73.19 $\pm$ 1.28	61.60 $\pm$ 0.64	82.34 $\pm$ 0.63	47.83 $\pm$ 0.63	62.71 $\pm$ 0.53	64.50 $\pm$ 2.28	66.16 $\pm$ 1.46	58.86 $\pm$ 0.77	54.95 $\pm$ 0.74	48.70 $\pm$ 0.00	8.60
APPNP	80.87 $\pm$ 2.20	74.66 $\pm$ 0.66	87.32 $\pm$ 0.39	76.79 $\pm$ 0.71	63.30 $\pm$ 0.34	57.28 $\pm$ 0.80	78.18 $\pm$ 0.31	65.51 $\pm$ 0.25	54.42 $\pm$ 0.59	57.86 $\pm$ 0.00	4.50
GPRGNN	77.86 $\pm$ 2.42	70.69 $\pm$ 0.22	87.17 $\pm$ 0.12	71.37 $\pm$ 0.74	61.41 $\pm$ 0.79	58.05 $\pm$ 0.17	77.44 $\pm$ 0.39	66.16 $\pm$ 0.35	60.50 $\pm$ 0.48	64.18 $\pm$ 0.00	4.50
RobustGCN	73.31 $\pm$ 2.70	62.70 $\pm$ 1.63	82.55 $\pm$ 0.52	34.20 $\pm$ 0.98	63.86 $\pm$ 0.50	52.26 $\pm$ 0.38	65.76 $\pm$ 0.76	58.02 $\pm$ 0.68	55.38 $\pm$ 0.64	50.41 $\pm$ 0.00	9.10
GCORN	74.29 $\pm$ 2.02	62.23 $\pm$ 0.89	83.01 $\pm$ 0.54	36.32 $\pm$ 1.47	55.29 $\pm$ 2.25	67.74 $\pm$ 2.32	59.00 $\pm$ 0.87	53.73 $\pm$ 0.63	46.66 $\pm$ 0.40	38.91 $\pm$ 0.00	9.70
NoisyGCN	72.82 $\pm$ 2.34	61.55 $\pm$ 0.85	83.77 $\pm$ 0.61	30.49 $\pm$ 1.29	55.28 $\pm$ 2.89	<b>68.01 <math>\pm</math> 2.17</b>	65.27 $\pm$ 0.68	58.19 $\pm$ 0.68	56.79 $\pm$ 0.15	41.32 $\pm$ 0.00	9.20
GRAND	78.60 $\pm$ 1.34	70.22 $\pm$ 1.22	83.88 $\pm$ 0.36	66.75 $\pm$ 0.42	<b>64.32 <math>\pm</math> 0.38</b>	65.06 $\pm$ 1.18	72.45 $\pm$ 0.62	65.91 $\pm$ 0.14	<b>61.17 <math>\pm</math> 0.27</b>	57.13 $\pm$ 0.00	4.40
SoftmedianGDC	78.66 $\pm$ 2.83	70.48 $\pm$ 0.53	86.61 $\pm$ 0.26	77.43 $\pm$ 0.34	59.22 $\pm$ 0.67	66.16 $\pm$ 0.10	77.78 $\pm$ 0.43	—	—	—	4.57
EvenNet	78.41 $\pm$ 2.03	68.86 $\pm$ 1.29	84.90 $\pm$ 0.17	55.20 $\pm$ 1.07	<b>64.18 <math>\pm</math> 0.86</b>	57.46 $\pm$ 1.19	69.13 $\pm$ 0.28	60.29 $\pm$ 0.29	58.03 $\pm$ 0.59	48.64 $\pm$ 0.00	6.10
ElasticGNN	73.74 $\pm$ 1.80	63.01 $\pm$ 1.67	82.94 $\pm$ 0.51	55.26 $\pm$ 0.68	56.32 $\pm$ 2.31	53.90 $\pm$ 3.37	68.84 $\pm$ 0.67	63.10 $\pm$ 0.35	54.48 $\pm$ 0.63	55.27 $\pm$ 0.00	8.40
GNNGuard	82.84 $\pm$ 1.14	<b>75.08 <math>\pm</math> 0.46</b>	88.73 $\pm$ 0.13	<b>80.66 <math>\pm</math> 0.59</b>	60.95 $\pm$ 1.04	56.83 $\pm$ 0.61	<b>82.93 <math>\pm</math> 0.16</b>	<b>74.64 <math>\pm</math> 0.23</b>	<b>73.27 <math>\pm</math> 0.15</b>	<b>71.25 <math>\pm</math> 0.00</b>	2.80
RUNG	<b>83.64 <math>\pm</math> 1.65</b>	73.46 $\pm$ 0.87	<b>89.78 <math>\pm</math> 0.14</b>	79.95 $\pm$ 0.23	63.93 $\pm$ 0.74	55.34 $\pm$ 0.80	—	—	—	—	3.50

Table 30: Accuracy under the inductive/evasion setting against structural attack. (ptb\_rate=0.2, atk\_emb=BoW, def\_emb=RoBERTa)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv	Avg Rank
GCN	73.31 $\pm$ 2.11	62.70 $\pm$ 0.59	82.92 $\pm$ 0.52	31.54 $\pm$ 0.71	60.60 $\pm$ 0.89	65.16 $\pm$ 2.29	65.81 $\pm$ 0.48	57.37 $\pm$ 0.97	55.09 $\pm$ 0.23	45.28 $\pm$ 0.00	10.50
GAT	73.43 $\pm$ 1.59	63.64 $\pm$ 0.90	83.60 $\pm$ 0.60	49.65 $\pm$ 1.12	62.86 $\pm$ 1.37	61.50 $\pm$ 1.18	69.08 $\pm$ 1.89	58.25 $\pm$ 0.89	54.77 $\pm$ 1.09	54.77 $\pm$ 0.00	9.10
APPNP	81.06 $\pm$ 2.31	70.43 $\pm$ 0.70	89.30 $\pm$ 0.18	76.53 $\pm$ 0.48	63.11 $\pm$ 1.23	55.37 $\pm$ 0.72	79.28 $\pm$ 0.40	65.14 $\pm$ 0.64	58.17 $\pm$ 0.46	58.06 $\pm$ 0.00	6.10
GPRGNN	79.27 $\pm$ 2.79	69.96 $\pm$ 1.82	89.47 $\pm$ 0.45	74.38 $\pm$ 0.68	60.85 $\pm$ 0.58	56.34 $\pm$ 0.35	81.30 $\pm$ 0.50	66.96 $\pm$ 0.62	62.63 $\pm$ 0.94	68.73 $\pm$ 0.00	6.20
RobustGCN	73.49 $\pm$ 1.80	60.97 $\pm$ 1.02	82.15 $\pm$ 0.24	36.85 $\pm$ 1.66	64.68 $\pm$ 0.32	57.59 $\pm$ 0.97	64.25 $\pm$ 0.40	56.74 $\pm$ 0.78	51.85 $\pm$ 2.42	51.90 $\pm$ 0.00	10.80
GCORN	73.49 $\pm$ 2.22	63.48 $\pm$ 0.66	83.98 $\pm$ 0.55	34.97 $\pm$ 0.79	58.11 $\pm$ 0.58	<b>68.09 <math>\pm</math> 1.86</b>	59.00 $\pm$ 1.36	53.80 $\pm$ 0.52	46.60 $\pm$ 0.17	41.82 $\pm$ 0.00	10.60
NoisyGCN	73.55 $\pm$ 1.35	62.59 $\pm$ 1.28	83.02 $\pm$ 0.48	32.14 $\pm$ 0.77	60.85 $\pm$ 0.70	65.44 $\pm$ 0.58	57.76 $\pm$ 1.11	55.09 $\pm$ 0.55	45.43 $\pm$ 0.00	9.60	—
GRAND	79.89 $\pm$ 0.80	70.74 $\pm$ 1.58	86.41 $\pm$ 0.52	71.86 $\pm$ 0.61	64.68 $\pm$ 0.32	63.44 $\pm$ 0.81	76.87 $\pm$ 0.18	<b>69.54 <math>\pm</math> 0.14</b>	<b>64.94 <math>\pm</math> 0.23</b>	69.80 $\pm$ 0.00	4.70
SoftmedianGDC	80.01 $\pm$ 2.53	70.53 $\pm$ 1.02	87.36 $\pm$ 0.26	81.43 $\pm$ 0.61	62.85 $\pm$ 0.61	63.10 $\pm$ 0.18	79.89 $\pm$ 0.51	—	—	—	5.71
EvenNet	79.33 $\pm$ 1.30	71.47 $\pm$ 0.64	87.53 $\pm$ 0.45	63.02 $\pm$ 0.33	66.64 $\pm$ 1.09	61.98 $\pm$ 2.69	72.86 $\pm$ 0.14	61.93 $\pm$ 0.38	59.79 $\pm$ 0.47	53.73 $\pm$ 0.00	6.00
ElasticGNN	74.91 $\pm$ 1.09	61.96 $\pm$ 1.03	84.22 $\pm$ 0.74	57.77 $\pm$ 1.03	57.10 $\pm$ 4.69	54.11 $\pm$ 1.06	70.63 $\pm$ 0.73	62.71 $\pm$ 0.60	54.45 $\pm$ 0.70	54.50 $\pm$ 0.00	9.90
GNNGuard	83.64 $\pm$ 1.94	74.03 $\pm$ 0.53	90.14 $\pm$ 0.44	83.03 $\pm$ 0.53	64.07 $\pm$ 1.19	58.58 $\pm$ 0.33	<b>84.64 <math>\pm</math> 0.49</b>	63.04 $\pm$ 0.71	63.07 $\pm$ 0.60	<b>71.16 <math>\pm</math> 0.00</b>	3.60
RUNG	<b>84.56 <math>\pm</math> 1.52</b>	73.30 $\pm$ 1.19	<b>90.72 <math>\pm</math> 0.25</b>	82.73 $\pm$ 0.78	<b>67.12 <math>\pm</math> 0.56</b>	58.11 $\pm$ 0.75	—	—	—	—	3.17
Guardual	83.95 $\pm$ 1.63	<b>74.71 <math>\pm</math> 1.22</b>	89.53 $\pm$ 0.39	<b>83.07 <math>\pm</math> 0.55</b>	65.40 $\pm$ 0.05	61.64 $\pm$ 0.46	<b>84.85 <math>\pm</math> 0.29</b>	<b>73.78 <math>\pm</math> 0.44</b>	<b>71.30 <math>\pm</math> 0.39</b>	<b>72.15 <math>\pm</math> 0.00</b>	2.10

Table 31: Accuracy under the transductive/poisoning setting against structural attack. (ptb\_rate=0.2, atk\_emb=BoW, def\_emb=BoW)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv	Avg Rank
GCN	64.56 $\pm$ 1.14	71.31 $\pm$ 0.44	77.59 $\pm$ 0.23	79.96 $\pm$ 0.35	63.87 $\pm$ 0.17	52.28 $\pm$ 1.42	64.39 $\pm$ 0.33	8.71	—	—	—
GAT	60.78 $\pm$ 2.21	70.55 $\pm$ 1.26	75.51 $\pm$ 0.28	78.55 $\pm$ 0.83	70.54 $\pm$ 1.75	53.49 $\pm$ 2.11	71.18 $\pm$ 1.21	8.57	—	—	—
APPNP	70.47 $\pm$ 1.21	<b>72.32 <math>\pm</math> 0.77</b>	83.61 $\pm$ 0.19	81.74 $\pm$ 0.78	79.02 $\pm$ 1.30	64.33 $\pm$ 0.25	<b>69.21 <math>\pm</math> 1.36</b>	73.49 $\pm$ 0.69	4.14	—	—
GPRGNN	71.45 $\pm$ 2.32	71.61 $\pm$ 0.99	81.74 $\pm$ 0.78	79.02 $\pm$ 1.30	69.35 $\pm$ 0.42	54.73 $\pm$ 0.79	<b>74.60 <math>\pm</math> 0.07</b>	—	—	—	—
RobustGCN	59.94 $\pm$ 0.31	70.16 $\pm$ 1.81	69.61 $\pm$ 0.35	<b>80.84 <math>\pm</math> 0.20</b>	63.56 $\pm$ 0.60	52.19 $\pm$ 0.21	63.72 $\pm$ 2.69	12.00	—	—	—
GCORN	47.92 $\pm$ 3.14	69.75 $\pm$ 0.80	55.58 $\pm$ 0.59	77.62 $\pm$ 0.41	63.77 $\pm$ 0.14	52.92 $\pm$ 0.27	58.59 $\pm$ 0.09	13.71	—	—	—
NoisyGCN	63.74 $\pm$ 0.44	71.87 $\pm$ 0.15	76.88 $\pm$ 0.76	79.79 $\pm$ 0.34	63.87 $\pm$ 0.17	52.21 $\pm$ 1.42	63.14 $\pm$ 1.96	9.86	—	—	—
GRAND	64.05 $\pm$ 3.51	72.22 $\pm$ 0.98	77.90 $\pm$ 1.65	79.98 $\pm$ 0.35	63.65 $\pm$ 0.13	55.05 $\pm$ 0.92	63.92 $\pm$ 0.06	7.57	—	—	—
SoftmedianGDC	69.65 $\pm$ 0.71	71.85 $\pm$ 0.73	80.65 $\pm$ 1.03	73.97 $\pm$ 0.50	65.30 $\pm$ 0.14	53.19 $\pm$ 1.01	72.79 $\pm$ 0.14	8.71	—	—	—
EvenNet	<b>72.28 <math>\pm</math> 1.98</b>	69.78 $\pm$ 0.66	<b>85.63 <math>\pm</math> 0.23</b>	80.79 $\pm$ 0.61	<b>79.57 <math>\pm</math> 0.84</b>	<b>66.64 <math>\pm</math> 0.55</b>	<b>77.04 <math>\pm</math> 0.20</b>	3.14	—	—	—
ElasticGNN	64.37 $\pm$ 1.89	71.64 $\pm$ 0.57	78.75 $\pm$ 0.68	79.90 $\pm$ 0.50	63.81 $\pm$ 0.26	51.76 $\pm$ 1.63	66.72 $\pm$ 0.80	9.14	—	—	—
GNNGuard	65.68 $\pm$ 0.36	70.81 $\pm$ 0.51	81.63 $\pm$ 0.39	77.45 $\pm$ 0.28	64.52 $\pm$ 0.62	54.60 $\pm$ 0.33	69.35 $\pm$ 0.79	7.57	—	—	—
RUNG	68.60 $\pm$ 2.18	71.14 $\pm$ 0.38	82.46 $\pm$ 0.27	78.20 $\pm$ 0.38	63.89 $\pm$ 0.14	52.37 $\pm$ 1.16	—	7.83	—	—	—
Cosine-GCN	65.13 $\pm$ 0.98	69.01 $\pm$ 1.45	82.03 $\pm$ 0.45	77.07 $\pm$ 0.36	64.36 $\pm$ 0.59	54.32 $\pm$ 0.99	70.11 $\pm$ 1.38	8.71	—	—	—
Jaccard-GCN	61.21 $\pm$ 0.80	69.49 $\pm$ 0.61	74.76 $\pm$ 0.52	78.45 $\pm$ 0.51	64.23 $\pm$ 0.64	51.89 $\pm$ 0.65	63.40 $\pm$ 2.38	12.14	—	—	—
Stable	60.18 $\pm$ 4.17	67.02 $\pm$ 1.18	72.35 $\pm$ 2.72	77.58 $\pm$ 0.58	63.89 $\pm$ 0.23	55.77 $\pm$ 0.73	—	11.67	—	—	—
ProGNN	<b>76.57 <math>\pm</math> 1.96</b>	70.22 $\pm$ 0.54	—	—	—	—	—	6.00	—	—	—

Table 32: Accuracy under the transductive/poisoning setting against structural attack. (ptb\_rate=0.2, atk\_emb=BoW, def\_emb=MiniLM)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv	Avg Rank





<tbl\_r cells="12" ix="5" maxcspan="1" maxrspan="1" used

Table 33: Accuracy under the transductive/poisoning setting against structural attack. (ptb\_rate=0.2, atk\_emb=BoW, def\_emb=RoBERTa)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Avg Rank
GCN	69.57 $\pm$ 3.44	74.68 $\pm$ 0.41	78.44 $\pm$ 1.30	81.69 $\pm$ 0.37	63.86 $\pm$ 0.18	52.77 $\pm$ 0.72	68.51 $\pm$ 1.53	11.00
GAT	67.64 $\pm$ 2.71	73.86 $\pm$ 0.43	78.80 $\pm$ 1.09	81.44 $\pm$ 0.13	72.22 $\pm$ 1.78	56.14 $\pm$ 3.40	78.09 $\pm$ 0.41	9.86
APPNP	80.62 $\pm$ 0.26	75.01 $\pm$ 0.50	88.23 $\pm$ 0.27	82.87 $\pm$ 0.50	66.24 $\pm$ 0.09	59.59 $\pm$ 0.21	83.73 $\pm$ 0.12	3.57
GPRGNN	79.93 $\pm$ 1.21	75.11 $\pm$ 0.59	87.36 $\pm$ 0.55	82.32 $\pm$ 0.38	73.19 $\pm$ 6.80	68.58 $\pm$ 0.59	82.70 $\pm$ 0.13	3.71
RobustGCN	66.44 $\pm$ 3.26	74.19 $\pm$ 0.28	64.78 $\pm$ 1.59	81.96 $\pm$ 0.26	63.64 $\pm$ 0.20	51.28 $\pm$ 0.46	64.55 $\pm$ 2.99	12.86
GCORN	67.65 $\pm$ 0.86	74.68 $\pm$ 0.96	61.51 $\pm$ 0.48	80.77 $\pm$ 0.40	63.88 $\pm$ 0.17	49.40 $\pm$ 0.33	62.32 $\pm$ 0.67	13.29
NoisyGCN	71.57 $\pm$ 1.42	74.57 $\pm$ 0.94	77.73 $\pm$ 0.92	81.61 $\pm$ 0.47	63.75 $\pm$ 0.21	53.08 $\pm$ 3.68	70.33 $\pm$ 1.22	11.29
GRAND	80.05 $\pm$ 0.76	<b>75.81 <math>\pm</math> 0.74</b>	83.73 $\pm$ 0.50	83.23 $\pm$ 0.45	63.93 $\pm$ 0.20	57.16 $\pm$ 0.40	77.99 $\pm$ 0.55	5.43
SoftmedianGDC	78.00 $\pm$ 0.44	73.14 $\pm$ 0.42	85.85 $\pm$ 0.43	79.78 $\pm$ 0.56	64.38 $\pm$ 0.35	57.77 $\pm$ 0.51	82.63 $\pm$ 0.26	9.43
EvenNet	80.82 $\pm$ 1.22	75.02 $\pm$ 0.16	<b>88.86 <math>\pm</math> 0.10</b>	<b>83.34 <math>\pm</math> 0.81</b>	<b>81.09 <math>\pm</math> 0.20</b>	<b>68.71 <math>\pm</math> 0.46</b>	<b>84.10 <math>\pm</math> 0.05</b>	1.57
ElasticGNN	75.02 $\pm$ 0.48	73.87 $\pm$ 0.19	82.05 $\pm$ 0.74	82.28 $\pm$ 0.17	63.84 $\pm$ 0.14	52.56 $\pm$ 1.83	70.57 $\pm$ 1.78	10.43
GNNGuard	79.93 $\pm$ 1.52	73.99 $\pm$ 0.62	86.95 $\pm$ 0.06	82.45 $\pm$ 0.75	66.80 $\pm$ 0.29	60.06 $\pm$ 0.46	83.48 $\pm$ 0.13	5.14
RUNG	79.23 $\pm$ 1.25	<b>75.45 <math>\pm</math> 0.88</b>	87.55 $\pm$ 0.45	81.80 $\pm$ 0.23	63.90 $\pm$ 0.23	54.61 $\pm$ 1.40	—	6.83
Cosine-GCN	79.88 $\pm$ 0.66	73.10 $\pm$ 1.19	86.75 $\pm$ 0.27	81.91 $\pm$ 0.71	66.65 $\pm$ 0.35	58.18 $\pm$ 0.38	83.54 $\pm$ 0.23	7.00
Jaccard-GCN	69.82 $\pm$ 1.49	72.47 $\pm$ 0.45	75.94 $\pm$ 1.43	81.05 $\pm$ 0.47	63.50 $\pm$ 0.49	49.56 $\pm$ 2.21	63.00 $\pm$ 4.95	14.29
Stable	78.28 $\pm$ 1.45	72.46 $\pm$ 0.66	77.66 $\pm$ 1.31	80.25 $\pm$ 0.96	63.93 $\pm$ 0.20	53.32 $\pm$ 2.22	—	12.00
ProGNN	<b>83.57 <math>\pm</math> 0.79</b>	74.98 $\pm$ 0.38	—	—	—	—	—	3.50

Table 34: Accuracy under the inductive/evasion setting against textual attack. (ptb\_rate=0.4, def\_emb=BoW)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Avg Rank
GCN	82.72 $\pm$ 1.65	60.66 $\pm$ 0.77	82.00 $\pm$ 1.30	<b>78.50 <math>\pm</math> 0.21</b>	61.16 $\pm$ 0.92	59.70 $\pm$ 1.08	78.01 $\pm$ 0.45	78.65 $\pm$ 0.25	4.38
GAT	82.90 $\pm$ 2.20	61.76 $\pm$ 2.50	83.05 $\pm$ 0.69	71.54 $\pm$ 1.13	63.21 $\pm$ 1.40	60.60 $\pm$ 1.30	<b>79.55 <math>\pm</math> 1.09</b>	82.05 $\pm$ 0.30	3.25
APPNP	80.07 $\pm$ 0.94	38.35 $\pm$ 1.03	67.82 $\pm$ 0.80	50.23 $\pm$ 0.54	60.42 $\pm$ 0.46	55.08 $\pm$ 0.21	74.79 $\pm$ 0.41	73.52 $\pm$ 0.54	9.88
GPRGNN	74.11 $\pm$ 2.38	40.39 $\pm$ 1.58	73.55 $\pm$ 2.15	54.82 $\pm$ 0.44	59.80 $\pm$ 1.14	58.65 $\pm$ 0.89	67.48 $\pm$ 0.93	71.34 $\pm$ 1.77	10.12
RobustGCN	<b>84.01 <math>\pm</math> 2.26</b>	<b>63.79 <math>\pm</math> 2.22</b>	82.16 $\pm$ 0.80	74.60 $\pm$ 0.52	<b>65.70 <math>\pm</math> 0.16</b>	56.33 $\pm$ 0.87	77.49 $\pm$ 0.31	78.39 $\pm$ 0.17	3.88
GCORN	77.31 $\pm$ 1.63	54.49 $\pm$ 0.64	77.09 $\pm$ 0.83	76.35 $\pm$ 0.19	62.02 $\pm$ 0.47	<b>65.51 <math>\pm</math> 0.65</b>	76.46 $\pm$ 0.35	74.52 $\pm$ 0.75	5.75
NoisyGCN	83.27 $\pm$ 1.35	59.93 $\pm$ 0.45	81.68 $\pm$ 1.24	78.39 $\pm$ 0.09	60.52 $\pm$ 0.61	61.32 $\pm$ 0.64	78.13 $\pm$ 0.31	79.03 $\pm$ 0.16	3.75
GRAND	75.77 $\pm$ 2.22	47.96 $\pm$ 1.48	78.54 $\pm$ 1.72	73.77 $\pm$ 0.24	64.64 $\pm$ 0.38	62.69 $\pm$ 2.23	75.60 $\pm$ 0.91	74.99 $\pm$ 0.10	6.25
SoftmedianGDC	70.54 $\pm$ 2.01	47.23 $\pm$ 0.27	70.63 $\pm$ 1.04	47.05 $\pm$ 0.39	59.16 $\pm$ 0.56	59.10 $\pm$ 0.36	61.46 $\pm$ 0.46	—	10.71
EvenNet	79.95 $\pm$ 0.74	52.72 $\pm$ 0.90	81.17 $\pm$ 1.09	74.94 $\pm$ 0.61	62.45 $\pm$ 1.25	59.61 $\pm$ 0.40	75.90 $\pm$ 0.49	<b>82.24 <math>\pm</math> 0.26</b>	5.50
ElasticGNN	82.96 $\pm$ 2.05	58.73 $\pm$ 1.53	<b>83.07 <math>\pm</math> 0.31</b>	69.59 $\pm$ 0.87	64.34 $\pm$ 1.24	59.96 $\pm$ 0.59	77.40 $\pm$ 0.29	78.73 $\pm$ 0.37	4.25
GNNGuard	76.81 $\pm$ 1.51	37.62 $\pm$ 0.34	66.37 $\pm$ 0.47	43.70 $\pm$ 0.32	58.91 $\pm$ 0.97	51.93 $\pm$ 0.65	68.78 $\pm$ 1.17	66.70 $\pm$ 0.63	11.75
RUNG	65.50 $\pm$ 4.14	43.26 $\pm$ 0.90	64.24 $\pm$ 1.08	48.90 $\pm$ 0.38	61.20 $\pm$ 0.67	55.69 $\pm$ 0.63	—	—	10.83

Table 35: Accuracy under the inductive/evasion setting against textual attack. (ptb\_rate=0.4, def\_emb=Minilm)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Avg Rank
GCN	80.01 $\pm$ 2.22	58.52 $\pm$ 0.91	78.25 $\pm$ 0.82	67.45 $\pm$ 0.54	59.01 $\pm$ 0.93	<b>64.33 <math>\pm</math> 0.20</b>	76.48 $\pm$ 0.33	73.66 $\pm$ 1.11	3.38
GAT	81.98 $\pm$ 1.09	58.62 $\pm$ 1.81	<b>78.66 <math>\pm</math> 3.94</b>	61.65 $\pm$ 1.15	63.18 $\pm$ 1.19	62.65 $\pm$ 1.46	<b>77.01 <math>\pm</math> 0.75</b>	72.78 $\pm$ 3.41	3.00
APPNP	58.73 $\pm$ 0.23	40.23 $\pm$ 1.84	66.51 $\pm$ 0.64	46.68 $\pm$ 0.30	58.83 $\pm$ 1.19	53.07 $\pm$ 0.32	66.07 $\pm$ 1.04	54.41 $\pm$ 0.26	10.25
GPRGNN	71.77 $\pm$ 2.13	43.16 $\pm$ 1.53	69.84 $\pm$ 0.66	46.53 $\pm$ 0.28	57.88 $\pm$ 1.54	53.98 $\pm$ 0.31	65.12 $\pm$ 0.85	54.71 $\pm$ 0.33	9.25
RobustGCN	<b>82.29 <math>\pm</math> 1.29</b>	<b>60.61 <math>\pm</math> 1.97</b>	78.15 $\pm$ 0.26	<b>69.83 <math>\pm</math> 1.37</b>	63.67 $\pm$ 0.51	55.63 $\pm$ 0.22	<b>77.73 <math>\pm</math> 0.41</b>	73.08 $\pm$ 0.67	2.88
GCORN	79.21 $\pm$ 0.53	54.70 $\pm$ 0.84	72.06 $\pm$ 0.70	62.45 $\pm$ 1.18	57.51 $\pm$ 0.46	64.15 $\pm$ 0.47	74.47 $\pm$ 0.50	63.62 $\pm$ 0.48	6.00
NoisyGCN	79.46 $\pm$ 1.43	58.41 $\pm$ 1.03	78.50 $\pm$ 0.83	<b>67.81 <math>\pm</math> 1.22</b>	58.85 $\pm$ 1.30	<b>64.67 <math>\pm</math> 0.76</b>	76.24 $\pm$ 0.48	74.12 $\pm$ 0.75	3.38
GRAND	68.08 $\pm$ 2.39	46.92 $\pm$ 1.41	69.79 $\pm$ 0.40	47.50 $\pm$ 0.37	<b>64.58 <math>\pm</math> 0.98</b>	60.92 $\pm$ 0.21	69.08 $\pm$ 0.62	55.84 $\pm$ 0.50	7.38
SoftmedianGDC	68.57 $\pm$ 0.63	51.15 $\pm$ 0.39	69.32 $\pm$ 0.74	54.46 $\pm$ 1.76	51.94 $\pm$ 0.42	—	63.31 $\pm$ 3.21	—	9.17
EvenNet	75.46 $\pm$ 1.34	48.33 $\pm$ 1.85	74.37 $\pm$ 0.68	51.86 $\pm$ 1.37	59.58 $\pm$ 0.73	56.55 $\pm$ 0.36	72.73 $\pm$ 0.44	<b>75.15 <math>\pm</math> 1.57</b>	6.00
ElasticGNN	76.75 $\pm$ 1.96	56.11 $\pm$ 1.89	<b>78.57 <math>\pm</math> 0.64</b>	51.80 $\pm$ 0.97	61.45 $\pm$ 1.02	60.19 $\pm$ 0.60	75.86 $\pm$ 0.73	60.90 $\pm$ 1.07	5.38
GNNGuard	52.34 $\pm$ 0.57	39.66 $\pm$ 0.56	65.82 $\pm$ 0.39	45.25 $\pm$ 0.34	51.53 $\pm$ 0.21	51.53 $\pm$ 0.77	56.90 $\pm$ 0.51	43.72 $\pm$ 0.22	12.50
RUNG	55.78 $\pm$ 0.63	39.76 $\pm$ 1.03	66.60 $\pm$ 0.34	46.45 $\pm$ 0.39	57.72 $\pm$ 1.00	53.22 $\pm$ 1.01	—	—	11.17

Table 36: Accuracy under the inductive/evasion setting against textual attack. (ptb\_rate=0.4, def\_emb=RoBERTa)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Avg Rank
GCN	81.12 $\pm$ 1.61	59.14 $\pm$ 0.96	80.13 $\pm$ 0.39	<b>73.27 <math>\pm</math> 0.93</b>	63.57 $\pm$ 1.47	65.42 $\pm$ 0.34	78.68 $\pm$ 0.22	74.01 $\pm$ 0.33	2.50
GAT	81.49 $\pm$ 1.58	57.84 $\pm$ 3.32	<b>81.94 <math>\pm</math> 0.97</b>	67.54 $\pm$ 7.56	64.29 $\pm$ 1.30	63.29 $\pm$ 0.12	<b>79.10 <math>\pm</math> 1.36</b>	<b>79.17 <math>\pm</math> 0.80</b>	2.62
APPNP	59.29 $\pm$ 1.84	42.58 $\pm$ 0.96	67.57 $\pm$ 0.55	48.21 $\pm$ 0.37	58.35 $\pm$ 1.28	55.12 $\pm$ 0.72	68.43 $\pm$ 0.51	54.96 $\pm$ 0.91	10.25
GPRGNN	66.30 $\pm$ 1.54	41.80 $\pm$ 0.27	67.43 $\pm$ 0.71	48.40 $\pm$ 0.79	57.39 $\pm$ 1.73	54.69 $\pm$ 0.37	62.84 $\pm$ 1.09	55.86 $\pm$ 2.28	10.25
RobustGCN	<b>83.03 <math>\pm</math> 2.11</b>	<b>60.24 <math>\pm</math> 1.07</b>	79.07 $\pm$ 1.13	69.81 $\pm$ 2.56	<b>64.51 <math>\pm</math> 0.57</b>	60.51 $\pm$ 0.60	78.51 $\pm$ 0.10	68.58 $\pm$ 1.45	3.25
GCORN	79.21 $\pm$ 1.28	56.06 $\pm$ 0.74	74.51 $\pm$ 0.58	66.75 $\pm$ 1.32	58.80 $\pm$ 1.18	64.97 $\pm$ 0.42	76.91 $\pm$ 0.67	64.81 $\pm$ 0.94	5.50
NoisyGCN	80.75 $\pm$ 1.43	<b>59.72 <math>\pm</math> 1.91</b>	79.97 $\pm$ 0.64	<b>73.25 <math>\pm</math> 0.97</b>	62.85 $\pm$ 1.84	<b>65.60 <math>\pm</math> 0.63</b>	<b>78.74 <math>\pm</math> 0.30</b>	73.90 $\pm$ 0.53	2.88
GRAND	69.43 $\pm$ 2.72	46.08 $\pm$ 1.56	72.62 $\pm$ 0.75	52.44 $\pm$ 0.50	<b>66.05 <math>\pm</math> 1.24</b>	63.42 $\pm$ 1.38	74.01 $\pm$ 0.47	57.19 $\pm$ 1.09	6.38
SoftmedianGDC	65.99 $\pm$ 0.31								

Table 37: Accuracy under the transductive/poisoning setting against textual attack. (ptb\_rate=0.8, def\_emb=BoW)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Avg Rank
GCN	83.63 $\pm$ 0.61	65.20 $\pm$ 1.19	85.16 $\pm$ 0.09	79.85 $\pm$ 0.25	65.81 $\pm$ 0.29	61.18 $\pm$ 0.61	79.69 $\pm$ 0.33	6.14
GAT	83.74 $\pm$ 0.12	65.54 $\pm$ 0.90	84.98 $\pm$ 0.05	79.05 $\pm$ 0.97	65.55 $\pm$ 0.34	60.92 $\pm$ 0.30	80.52 $\pm$ 0.16	6.86
APPNP	<b>84.59 <math>\pm</math> 0.69</b>	66.58 $\pm$ 1.11	86.48 $\pm$ 0.28	79.88 $\pm$ 0.49	65.65 $\pm$ 0.38	60.60 $\pm$ 0.17	<b>80.72 <math>\pm</math> 0.13</b>	3.57
GPRGNN	84.43 $\pm$ 0.45	66.93 $\pm$ 1.32	85.75 $\pm$ 0.07	78.83 $\pm$ 0.14	65.56 $\pm$ 0.44	59.83 $\pm$ 0.40	80.12 $\pm$ 0.22	6.43
RobustGCN	83.65 $\pm$ 0.48	64.60 $\pm$ 1.05	85.53 $\pm$ 0.16	<b>80.39 <math>\pm</math> 0.37</b>	65.59 $\pm$ 0.12	59.26 $\pm$ 0.31	80.62 $\pm$ 0.10	6.00
GCORN	74.71 $\pm$ 1.80	64.35 $\pm$ 1.03	83.32 $\pm$ 0.19	77.63 $\pm$ 0.41	64.79 $\pm$ 0.51	<b>65.30 <math>\pm</math> 0.16</b>	77.94 $\pm$ 0.17	10.86
NoisyGCN	83.13 $\pm$ 0.70	65.36 $\pm$ 0.83	85.26 $\pm$ 0.12	79.77 $\pm$ 0.26	65.81 $\pm$ 0.35	63.27 $\pm$ 0.65	80.09 $\pm$ 0.09	6.00
GRAND	83.93 $\pm$ 0.30	67.70 $\pm$ 0.68	<b>86.58 <math>\pm</math> 0.25</b>	79.46 $\pm$ 0.46	65.53 $\pm$ 0.25	62.91 $\pm$ 0.08	80.28 $\pm$ 0.15	4.43
SoftmedianGDC	82.23 $\pm$ 0.88	60.27 $\pm$ 1.34	83.70 $\pm$ 0.04	70.02 $\pm$ 0.11	<b>66.09 <math>\pm</math> 0.31</b>	61.00 $\pm$ 0.58	77.21 $\pm$ 0.35	10.71
EvenNet	83.39 $\pm$ 1.02	59.98 $\pm$ 0.16	85.98 $\pm$ 0.17	80.07 $\pm$ 0.59	65.30 $\pm$ 0.46	<b>64.56 <math>\pm</math> 0.53</b>	80.19 $\pm$ 0.13	6.71
ElasticGNN	83.62 $\pm$ 0.53	<b>68.37 <math>\pm</math> 0.93</b>	86.39 $\pm$ 0.21	79.78 $\pm$ 0.14	65.53 $\pm$ 0.27	60.59 $\pm$ 0.38	80.52 $\pm$ 0.05	5.43
GNNGuard	79.23 $\pm$ 0.36	17.98 $\pm$ 0.47	85.14 $\pm$ 0.19	72.80 $\pm$ 0.55	64.37 $\pm$ 0.14	58.73 $\pm$ 0.30	78.93 $\pm$ 0.09	13.29
RUNG	82.88 $\pm$ 0.59	61.48 $\pm$ 1.49	85.01 $\pm$ 0.24	77.69 $\pm$ 0.51	65.28 $\pm$ 0.25	60.49 $\pm$ 0.11	—	10.67
Cosine-GCN	78.59 $\pm$ 0.51	59.74 $\pm$ 0.21	82.06 $\pm$ 0.30	72.68 $\pm$ 0.18	64.72 $\pm$ 0.11	58.66 $\pm$ 0.34	77.44 $\pm$ 0.22	14.86
Jaccard-GCN	79.17 $\pm$ 1.96	60.25 $\pm$ 0.58	82.36 $\pm$ 0.22	77.38 $\pm$ 0.36	64.94 $\pm$ 0.59	61.16 $\pm$ 0.27	77.72 $\pm$ 0.03	12.14
Stable	77.86 $\pm$ 0.62	61.11 $\pm$ 0.40	82.22 $\pm$ 0.24	77.51 $\pm$ 0.16	64.59 $\pm$ 0.22	62.95 $\pm$ 0.34	74.80 $\pm$ 0.35	12.57
ProGNN	76.08 $\pm$ 0.34	66.07 $\pm$ 0.84	—	—	—	—	—	10.50

Table 38: Accuracy under the transductive/poisoning setting against textual attack. (ptb\_rate=0.8, def\_emb=MiniLM)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Avg Rank
GCN	83.65 $\pm$ 1.28	69.52 $\pm$ 0.53	86.01 $\pm$ 0.31	80.65 $\pm$ 0.28	65.16 $\pm$ 0.58	65.36 $\pm$ 0.38	82.46 $\pm$ 0.07	8.14
GAT	84.33 $\pm$ 0.67	70.46 $\pm$ 0.75	85.38 $\pm$ 0.28	81.47 $\pm$ 0.53	65.29 $\pm$ 0.07	63.69 $\pm$ 0.43	82.45 $\pm$ 0.18	6.71
APPNP	84.17 $\pm$ 0.90	70.94 $\pm$ 0.62	87.35 $\pm$ 0.05	81.13 $\pm$ 0.16	65.70 $\pm$ 0.69	61.86 $\pm$ 0.27	<b>83.13 <math>\pm</math> 0.06</b>	4.57
GPRGNN	84.23 $\pm$ 0.56	70.35 $\pm$ 0.40	<b>87.47 <math>\pm</math> 0.40</b>	80.76 $\pm$ 0.25	64.76 $\pm$ 0.47	61.40 $\pm$ 0.48	82.31 $\pm$ 0.08	7.86
RobustGCN	<b>84.60 <math>\pm</math> 0.61</b>	69.77 $\pm$ 0.28	86.04 $\pm$ 0.21	81.41 $\pm$ 0.13	<b>65.98 <math>\pm</math> 0.23</b>	64.06 $\pm$ 0.39	82.81 $\pm$ 0.08	4.57
GCORN	82.66 $\pm$ 1.20	71.26 $\pm$ 0.77	85.30 $\pm$ 0.22	78.91 $\pm$ 0.10	65.72 $\pm$ 0.25	67.37 $\pm$ 0.13	82.41 $\pm$ 0.10	7.14
NoisyGCN	83.53 $\pm$ 0.75	68.72 $\pm$ 0.22	86.04 $\pm$ 0.38	80.35 $\pm$ 0.40	<b>65.87 <math>\pm</math> 0.63</b>	<b>67.86 <math>\pm</math> 0.28</b>	82.69 $\pm$ 0.10	6.57
GRAND	<b>84.85 <math>\pm</math> 0.55</b>	<b>71.60 <math>\pm</math> 0.90</b>	85.17 $\pm$ 0.24	78.54 $\pm$ 0.65	65.48 $\pm$ 0.19	65.24 $\pm$ 0.34	82.07 $\pm$ 0.07	7.14
SoftmedianGDC	84.22 $\pm$ 0.37	65.87 $\pm$ 1.06	85.78 $\pm$ 0.60	77.88 $\pm$ 0.43	65.45 $\pm$ 0.41	—	82.99 $\pm$ 0.10	9.33
EvenNet	84.09 $\pm$ 0.40	68.51 $\pm$ 0.60	86.90 $\pm$ 0.05	<b>82.00 <math>\pm</math> 0.34</b>	65.59 $\pm$ 0.18	61.70 $\pm$ 0.09	82.94 $\pm$ 0.18	6.43
ElasticGNN	84.37 $\pm$ 0.70	69.94 $\pm$ 0.80	87.28 $\pm$ 0.16	81.36 $\pm$ 0.11	65.17 $\pm$ 0.75	62.62 $\pm$ 0.17	<b>83.05 <math>\pm</math> 0.18</b>	5.71
GNNGuard	79.70 $\pm$ 0.25	17.98 $\pm$ 0.47	83.85 $\pm$ 0.35	76.18 $\pm$ 0.36	65.39 $\pm$ 0.92	58.95 $\pm$ 0.32	80.99 $\pm$ 0.15	14.29
RUNG	81.96 $\pm$ 0.79	67.09 $\pm$ 0.48	86.45 $\pm$ 0.13	80.41 $\pm$ 0.56	64.93 $\pm$ 0.04	61.32 $\pm$ 0.16	—	11.33
Cosine-GCN	79.45 $\pm$ 1.23	67.50 $\pm$ 1.22	82.62 $\pm$ 0.11	76.23 $\pm$ 0.31	65.10 $\pm$ 0.68	58.91 $\pm$ 0.29	80.64 $\pm$ 0.14	14.86
Jaccard-GCN	81.63 $\pm$ 1.34	67.09 $\pm$ 0.59	84.15 $\pm$ 0.25	80.11 $\pm$ 0.29	65.51 $\pm$ 0.16	66.19 $\pm$ 0.12	81.46 $\pm$ 0.17	10.43
Stable	81.62 $\pm$ 0.95	70.35 $\pm$ 0.66	84.06 $\pm$ 0.32	80.54 $\pm$ 0.58	65.35 $\pm$ 0.56	64.67 $\pm$ 0.22	—	9.67
ProGNN	81.88 $\pm$ 0.89	69.30 $\pm$ 0.58	—	—	—	—	—	11.50

Table 39: Accuracy under the transductive/poisoning setting against textual attack. (ptb\_rate=0.8, def\_emb=RoBERTa)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Avg Rank
GCN	84.29 $\pm$ 0.68	68.67 $\pm$ 0.21	86.38 $\pm$ 0.15	81.70 $\pm$ 0.11	66.56 $\pm$ 0.22	<b>63.80 <math>\pm</math> 0.27</b>	83.21 $\pm$ 0.13	8.14
GAT	84.97 $\pm$ 0.51	67.58 $\pm$ 1.60	85.54 $\pm$ 0.24	81.81 $\pm$ 0.30	66.37 $\pm$ 0.49	63.84 $\pm$ 0.48	83.31 $\pm$ 0.10	8.57
APPNP	85.03 $\pm$ 0.31	70.97 $\pm$ 0.35	<b>87.83 <math>\pm</math> 0.27</b>	81.39 $\pm$ 0.06	66.86 $\pm$ 0.66	63.65 $\pm$ 0.28	83.57 $\pm$ 0.23	5.43
GPRGNN	85.00 $\pm$ 0.59	70.00 $\pm$ 1.38	<b>87.82 <math>\pm</math> 0.47</b>	81.65 $\pm$ 0.07	66.76 $\pm$ 0.18	62.64 $\pm$ 0.46	83.53 $\pm$ 0.22	6.43
RobustGCN	84.28 $\pm$ 0.24	68.05 $\pm$ 0.23	86.28 $\pm$ 0.08	81.48 $\pm$ 0.16	65.89 $\pm$ 0.51	63.47 $\pm$ 0.12	83.00 $\pm$ 0.13	10.57
GCORN	83.40 $\pm$ 0.44	71.00 $\pm$ 1.12	85.69 $\pm$ 0.05	81.09 $\pm$ 0.37	66.58 $\pm$ 0.20	<b>67.71 <math>\pm</math> 0.11</b>	83.67 $\pm$ 0.03	7.29
NoisyGCN	84.29 $\pm$ 0.23	68.35 $\pm$ 1.19	86.41 $\pm$ 0.08	81.47 $\pm$ 0.21	66.40 $\pm$ 0.33	63.65 $\pm$ 0.26	83.31 $\pm$ 0.03	8.71
GRAND	<b>85.54 <math>\pm</math> 0.35</b>	71.40 $\pm$ 0.17	87.15 $\pm$ 0.21	<b>82.89 <math>\pm</math> 0.20</b>	<b>67.29 <math>\pm</math> 0.20</b>	<b>67.02 <math>\pm</math> 0.54</b>	<b>84.31 <math>\pm</math> 0.09</b>	2.14
SoftmedianGDC	83.33 $\pm$ 0.56	64.50 $\pm$ 0.98	86.25 $\pm$ 0.18	76.51 $\pm$ 0.37	66.10 $\pm$ 0.27	—	83.41 $\pm$ 0.08	12.67
EvenNet	84.96 $\pm$ 0.15	67.87 $\pm$ 0.59	87.64 $\pm$ 0.31	82.77 $\pm$ 0.27	66.78 $\pm$ 0.58	63.28 $\pm$ 1.04	83.63 $\pm$ 0.02	6.29
ElasticGNN	<b>85.25 <math>\pm</math> 0.81</b>	<b>72.02 <math>\pm</math> 0.50</b>	87.62 $\pm$ 0.09	82.33 $\pm$ 0.29	66.75 $\pm$ 0.34	63.88 $\pm$ 0.38	<b>83.99 <math>\pm</math> 0.05</b>	3.43
GNNGuard	80.23 $\pm$ 0.78	17.98 $\pm$ 0.47	84.35 $\pm$ 0.30	77.26 $\pm$ 0.38	66.88 $\pm$ 0.36	58.93 $\pm$ 0.69	82.19 $\pm$ 0.17	13.00
RUNG	83.37 $\pm$ 0.56	68.71 $\pm$ 0.64	87.37 $\pm$ 0.28	81.56 $\pm$ 0.69	65.95 $\pm$ 1.54	63.01 $\pm$ 0.08	—	9.83
Cosine-GCN	77.33 $\pm$ 0.56	66.41 $\pm$ 0.66	83.29 $\pm$ 0.23	77.03 $\pm$ 0.54	66.58 $\pm$ 0.70	57.61 $\pm$ 1.08	81.77 $\pm$ 0.18	14.43
Jaccard-GCN	82.36 $\pm$ 0.85	66.92 $\pm$ 0.91	84.95 $\pm$ 0.22	81.04 $\pm$ 0.27	66.87 $\pm$ 0.31	63.82 $\pm$ 0.19	82.50 $\pm$ 0.18	10.86
Stable	83.83 $\pm$ 0.35	71.31 $\pm$ 0.67	84.87 $\pm$ 0.12	80.68 $\pm$ 0.53	<b>66.88 <math>\pm</math> 0.18</b>	65.14 $\pm$ 0.38	—	7.67
ProGNN	82.48 $\pm$ 1.08	71.61 $\pm$ 2.06	—	—	—	—	—	8.00

### J.3 GRAPHLLM RESULTS

Results are in Tables 40, 41, 42, 43, 44, 45.

Table 40: GraphLLM’s clean results in the inductive/evasion setting.

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv
GraphGPT	81.06 ± 2.33	74.35 ± 2.51	94.14 ± 0.23	82.31 ± 1.31	66.93 ± 1.44	61.87 ± 0.52	85.34 ± 1.07	84.77 ± 0.34	85.80 ± 0.60	74.78
GraphGPT-noise	80.63 ± 2.24	74.56 ± 3.07	94.09 ± 0.33	82.32 ± 1.97	67.59 ± 0.67	60.89 ± 1.61	85.70 ± 0.72	83.55 ± 0.58	—	—
GraphGPT-noisetxt	66.79 ± 3.84	57.37 ± 5.20	86.15 ± 1.88	64.56 ± 2.94	—	—	—	—	—	—
LLaGA	86.29 ± 1.76	75.81 ± 0.63	90.23 ± 0.59	84.88 ± 1.60	68.06 ± 0.49	68.53 ± 1.34	85.96 ± 0.53	86.96 ± 0.21	90.00 ± 0.20	74.52
LLaGA-noise	86.04 ± 2.18	75.55 ± 0.57	90.55 ± 0.57	84.14 ± 0.49	68.46 ± 0.59	68.62 ± 0.77	85.81 ± 0.38	86.57 ± 0.50	—	—
LLaGA-noisefull	83.70 ± 2.23	75.29 ± 1.01	89.13 ± 0.59	83.15 ± 1.64	68.31 ± 0.87	68.91 ± 0.48	85.52 ± 0.34	86.60 ± 0.39	—	—
LLaGA-noisetxt	83.64 ± 1.66	73.72 ± 1.89	86.37 ± 3.70	80.89 ± 1.20	68.15 ± 0.34	67.99 ± 2.25	84.11 ± 1.22	84.27 ± 1.78	—	—
LLaGA-sim	83.76 ± 1.03	75.08 ± 0.96	90.27 ± 0.19	82.39 ± 2.59	67.40 ± 0.46	56.75 ± 1.04	85.48 ± 0.85	79.51 ± 0.04	—	—
SFT-auto	83.27 ± 2.34	74.56 ± 1.19	94.74 ± 0.24	86.40 ± 0.69	68.39 ± 1.04	66.61 ± 1.44	86.08 ± 0.50	84.58 ± 0.09	83.58 ± 0.11	—
SFT (w/o neighbor)	82.96 ± 1.65	74.19 ± 0.48	94.98 ± 0.18	87.30 ± 0.20	69.27 ± 0.40	62.70 ± 0.50	86.51 ± 0.80	80.78 ± 0.35	77.19 ± 0.21	76.83
SFT-neighbor	88.25 ± 1.29	76.17 ± 1.81	95.08 ± 0.51	87.75 ± 0.46	69.24 ± 0.54	66.74 ± 1.06	86.81 ± 0.77	87.51 ± 0.07	89.74 ± 0.14	77.84
SFT-noise	85.30 ± 2.58	75.65 ± 1.89	95.13 ± 0.27	87.78 ± 0.30	69.78 ± 0.88	67.33 ± 1.09	86.53 ± 0.67	87.52 ± 0.13	89.65 ± 0.11	77.93
SFT-noisefull	87.45 ± 1.87	78.00 ± 1.42	94.96 ± 0.19	86.59 ± 0.20	69.39 ± 0.54	67.13 ± 0.68	86.31 ± 0.88	86.76 ± 0.12	89.37 ± 0.07	77.35
SFT-noisetxt	86.29 ± 1.26	76.18 ± 1.34	94.96 ± 0.23	86.70 ± 0.43	69.59 ± 0.50	66.99 ± 0.69	86.12 ± 0.47	86.78 ± 0.12	89.37 ± 0.19	77.60
SFT-simp	88.19 ± 1.58	76.38 ± 1.33	95.00 ± 0.08	87.84 ± 0.29	69.37 ± 0.33	66.23 ± 1.30	86.61 ± 0.71	87.53 ± 0.32	89.71 ± 0.04	77.68
SFT-simf	85.30 ± 2.04	76.02 ± 1.96	95.16 ± 0.39	87.61 ± 0.49	69.66 ± 0.46	51.78 ± 1.14	86.72 ± 0.87	83.12 ± 0.18	—	—

Table 41: GraphLLM’s clean results in the transductive/poisoning setting.

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo
GraphGPT	68.97 ± 2.30	67.28 ± 1.56	92.13 ± 0.69	75.35 ± 0.16	63.21 ± 1.83	59.92 ± 0.06	83.46 ± 0.13	80.86 ± 0.55
GraphGPT-noise	69.71 ± 1.63	66.07 ± 0.06	91.87 ± 0.79	75.66 ± 0.12	55.44 ± 3.63	58.51 ± 0.58	83.09 ± 0.25	80.14 ± 0.89
LLaGA	78.94 ± 0.47	70.51 ± 2.98	85.60 ± 3.83	79.65 ± 1.15	61.56 ± 4.71	68.44 ± 1.39	83.55 ± 0.30	—
LLaGA-noise	79.53 ± 2.37	71.40 ± 1.76	84.71 ± 3.98	77.08 ± 1.59	66.65 ± 0.15	68.59 ± 0.37	82.94 ± 0.34	82.72 ± 0.52
LLaGA-noisefull	78.39 ± 3.82	71.82 ± 1.81	86.10 ± 1.46	78.39 ± 0.81	63.04 ± 4.14	68.18 ± 0.90	83.40 ± 0.42	83.40 ± 1.20
LLaGA-noisetxt	80.11 ± 3.95	71.26 ± 0.70	86.39 ± 1.17	79.60 ± 0.93	65.28 ± 2.14	67.31 ± 1.06	83.71 ± 0.22	84.38 ± 0.67
SFT (w/o neighbor)	75.45 ± 2.03	72.10 ± 0.94	93.93 ± 0.35	82.31 ± 0.49	66.53 ± 1.03	60.49 ± 1.02	84.78 ± 0.50	—
SFT-neighbor	79.40 ± 1.97	72.03 ± 2.03	93.80 ± 0.31	82.65 ± 0.83	66.85 ± 0.96	66.01 ± 0.57	85.72 ± 0.09	85.05 ± 0.57
SFT-noise	81.46 ± 2.39	71.88 ± 1.44	94.14 ± 0.35	83.12 ± 1.27	65.95 ± 1.18	66.02 ± 0.42	85.47 ± 0.12	—
SFT-noisefull	80.34 ± 1.32	70.79 ± 3.08	93.53 ± 0.49	82.86 ± 0.51	66.09 ± 1.83	66.17 ± 0.17	85.38 ± 0.13	—
SFT-noisetxt	76.92 ± 2.00	72.31 ± 0.49	93.46 ± 0.26	82.48 ± 0.12	66.23 ± 0.40	65.96 ± 0.74	85.07 ± 0.58	—
SFT-simp	78.80 ± 1.46	72.77 ± 1.22	94.06 ± 0.37	83.59 ± 0.61	66.37 ± 1.17	63.52 ± 2.06	85.54 ± 0.13	—

Table 42: GraphLLM’s results under the inductive/evasion setting against structural attacks (ptb\_rate=0.20).

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv
GraphGPT	77.68 ± 4.77	73.51 ± 0.72	93.62 ± 0.77	76.51 ± 5.85	65.31 ± 1.03	58.01 ± 0.74	81.59 ± 3.79	77.47 ± 3.32	76.12 ± 5.52	70.61
GraphGPT-noise	78.60 ± 5.05	73.98 ± 1.90	93.81 ± 0.44	78.87 ± 4.39	65.24 ± 1.37	58.01 ± 0.88	83.20 ± 2.69	—	—	—
LLaGA	75.21 ± 3.73	66.35 ± 1.25	87.38 ± 1.15	66.99 ± 5.07	66.33 ± 1.44	73.48 ± 5.39	81.24 ± 1.66	63.91 ± 4.11	62.32 ± 2.80	70.68
LLaGA-noise	76.26 ± 0.75	68.03 ± 1.57	88.29 ± 0.65	74.95 ± 1.94	67.73 ± 0.53	73.65 ± 6.02	82.34 ± 1.26	71.88 ± 0.63	—	—
LLaGA-noisefull	74.17 ± 2.05	66.35 ± 0.59	85.38 ± 0.84	62.40 ± 2.24	66.92 ± 1.73	75.16 ± 3.43	73.41 ± 2.06	66.48 ± 1.69	—	—
LLaGA-noisetxt	72.02 ± 1.26	65.41 ± 1.25	82.67 ± 0.57	51.17 ± 1.70	66.01 ± 1.31	74.44 ± 4.75	68.14 ± 0.39	57.96 ± 1.41	—	—
LLaGA-sim	81.92 ± 2.61	73.77 ± 0.48	89.43 ± 0.58	82.91 ± 0.46	68.21 ± 0.49	55.20 ± 1.68	85.57 ± 0.39	76.32 ± 0.42	73.99 ± 0.40	73.73
SFT-auto	82.59 ± 3.53	74.24 ± 2.14	92.29 ± 0.45	84.05 ± 0.69	66.93 ± 0.99	66.18 ± 2.74	85.54 ± 0.69	83.99 ± 0.14	83.62 ± 0.13	—
SFT-neighbor	82.35 ± 1.94	71.53 ± 1.86	94.73 ± 0.41	86.00 ± 1.40	68.25 ± 0.20	62.29 ± 3.31	85.98 ± 0.51	81.72 ± 0.34	82.39 ± 0.20	77.21
SFT-noise	79.52 ± 2.72	73.14 ± 0.36	94.51 ± 0.31	84.38 ± 0.60	68.45 ± 0.48	54.54 ± 2.95	85.31 ± 0.73	81.75 ± 0.12	—	77.49
SFT-noisefull	78.04 ± 0.85	70.17 ± 0.86	91.53 ± 0.55	82.07 ± 1.23	68.62 ± 1.01	62.39 ± 2.69	76.16 ± 2.83	79.00 ± 0.84	—	73.89
SFT-noisetxt	77.80 ± 2.51	68.18 ± 1.44	90.35 ± 0.51	79.98 ± 2.08	68.37 ± 0.72	62.07 ± 2.06	79.52 ± 2.36	78.20 ± 1.84	78.24 ± 1.07	74.38
SFT-simp	81.12 ± 2.77	72.26 ± 1.63	94.61 ± 0.17	86.39 ± 0.86	68.87 ± 0.36	64.27 ± 1.27	85.69 ± 0.23	81.63 ± 0.53	82.45 ± 0.42	77.11
SFT-simf	84.19 ± 2.72	75.29 ± 2.40	94.96 ± 0.36	86.00 ± 1.36	69.21 ± 0.60	64.88 ± 2.96	86.50 ± 0.89	81.23 ± 0.09	—	—

### J.4 RESULTS WITH DIFFERENT PERTURB RATIOS

In Tables 46, 47, 48 and 49, we present results against different attack ratios. In Figures 12 and 13, we further visualize the accuracy–perturbation curves across different datasets under structural and textual attacks in the inductive setting. The trends align with our observations in the main paper.

Table 43: GraphLLM’s results under the transductive/poisoning setting against structural attacks. (ptb\_rate=0.30)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo
GraphGPT	64.53 $\pm$ 1.81	64.91 $\pm$ 1.55	90.02 $\pm$ 2.33	76.28 $\pm$ 0.95	58.95 $\pm$ 0.69	45.73 $\pm$ 0.66	79.82 $\pm$ 0.38	73.58 $\pm$ 1.47
GraphGPT-noise	65.70 $\pm$ 0.76	64.46 $\pm$ 2.67	90.60 $\pm$ 1.77	75.06 $\pm$ 0.70	58.13 $\pm$ 0.94	45.78 $\pm$ 0.35	80.82 $\pm$ 0.30	74.02 $\pm$ 0.37
LLaGA	71.02 $\pm$ 4.70	69.41 $\pm$ 1.42	81.63 $\pm$ 1.58	76.70 $\pm$ 1.25	58.94 $\pm$ 1.72	45.07 $\pm$ 0.76	79.00 $\pm$ 0.72	72.54 $\pm$ 2.18
LLaGA-noise	72.39 $\pm$ 1.59	71.41 $\pm$ 1.61	82.91 $\pm$ 1.36	79.07 $\pm$ 1.79	55.43 $\pm$ 10.44	46.90 $\pm$ 0.70	80.16 $\pm$ 0.28	73.43 $\pm$ 1.48
LLaGA-noisefull	64.22 $\pm$ 2.80	68.69 $\pm$ 2.53	81.11 $\pm$ 0.64	74.85 $\pm$ 3.57	59.83 $\pm$ 0.96	45.29 $\pm$ 0.57	73.68 $\pm$ 1.58	73.14 $\pm$ 0.59
LLaGA-noisetxt	61.92 $\pm$ 9.72	68.12 $\pm$ 1.98	81.05 $\pm$ 1.11	76.37 $\pm$ 1.81	57.36 $\pm$ 5.04	45.63 $\pm$ 1.36	72.64 $\pm$ 2.52	73.04 $\pm$ 0.93
SFT-neighbor	69.65 $\pm$ 4.42	71.99 $\pm$ 1.28	92.33 $\pm$ 0.62	82.00 $\pm$ 1.18	62.48 $\pm$ 2.60	51.90 $\pm$ 1.02	82.82 $\pm$ 1.61	80.51 $\pm$ 0.51
SFT-noise	70.56 $\pm$ 2.34	71.56 $\pm$ 1.82	92.84 $\pm$ 0.71	82.73 $\pm$ 0.61	60.95 $\pm$ 4.73	51.25 $\pm$ 0.86	83.38 $\pm$ 0.60	81.25 $\pm$ 0.82
SFT-noisefull	71.48 $\pm$ 4.09	71.78 $\pm$ 0.71	92.35 $\pm$ 0.86	81.92 $\pm$ 0.56	63.90 $\pm$ 0.24	51.39 $\pm$ 0.47	83.71 $\pm$ 0.29	81.16 $\pm$ 0.52
SFT-noisetxt	71.10 $\pm$ 1.19	70.52 $\pm$ 3.10	92.02 $\pm$ 0.47	81.35 $\pm$ 0.71	60.77 $\pm$ 5.23	53.60 $\pm$ 0.32	83.69 $\pm$ 0.67	80.74 $\pm$ 0.63
SFT-simp	70.45 $\pm$ 5.39	70.89 $\pm$ 1.16	92.65 $\pm$ 1.18	82.82 $\pm$ 1.41	63.89 $\pm$ 0.23	51.73 $\pm$ 0.63	83.48 $\pm$ 0.83	80.76 $\pm$ 0.48

Table 44: GraphLLM’s results under the inductive/evasion setting against textual attacks. (ptb\_rate=0.40)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo
GraphGPT	66.48 $\pm$ 15.39	59.51 $\pm$ 13.47	71.21 $\pm$ 14.22	66.45 $\pm$ 17.17	63.38 $\pm$ 3.93	57.27 $\pm$ 3.57	70.53 $\pm$ 14.86	54.23 $\pm$ 2.63
GraphGPT-noise	51.66 $\pm$ 4.54	59.87 $\pm$ 13.78	70.36 $\pm$ 15.72	65.91 $\pm$ 18.43	62.99 $\pm$ 4.04	57.56 $\pm$ 2.34	70.26 $\pm$ 15.90	—
LLaGA	81.73 $\pm$ 1.76	61.65 $\pm$ 2.30	73.84 $\pm$ 1.08	61.14 $\pm$ 3.87	63.40 $\pm$ 3.54	68.83 $\pm$ 1.61	72.55 $\pm$ 1.70	77.58 $\pm$ 1.62
LLaGA-noise	80.88 $\pm$ 1.67	61.23 $\pm$ 3.46	70.82 $\pm$ 1.00	51.29 $\pm$ 1.55	59.44 $\pm$ 1.15	68.21 $\pm$ 2.07	71.15 $\pm$ 1.53	67.24 $\pm$ 1.28
LLaGA-noisefull	79.89 $\pm$ 2.13	61.65 $\pm$ 1.04	77.12 $\pm$ 0.21	63.40 $\pm$ 1.99	62.14 $\pm$ 2.41	68.59 $\pm$ 0.56	78.06 $\pm$ 0.94	74.93 $\pm$ 1.80
LLaGA-noisetxt	78.54 $\pm$ 0.93	59.19 $\pm$ 0.80	79.51 $\pm$ 1.73	67.92 $\pm$ 2.92	64.70 $\pm$ 1.35	69.10 $\pm$ 1.38	78.15 $\pm$ 0.69	77.00 $\pm$ 1.73
LLaGA-sim	52.21 $\pm$ 1.66	39.08 $\pm$ 0.77	65.68 $\pm$ 0.42	47.10 $\pm$ 0.36	55.12 $\pm$ 2.67	52.03 $\pm$ 0.71	60.24 $\pm$ 4.61	47.08 $\pm$ 0.58
SFT-auto	76.75 $\pm$ 1.76	65.36 $\pm$ 3.29	77.86 $\pm$ 1.38	77.83 $\pm$ 0.55	61.70 $\pm$ 0.84	65.68 $\pm$ 0.87	73.55 $\pm$ 0.83	77.96 $\pm$ 1.76
SFT-neighbor	75.65 $\pm$ 1.33	43.84 $\pm$ 2.14	72.27 $\pm$ 3.53	51.53 $\pm$ 1.24	64.33 $\pm$ 1.46	65.52 $\pm$ 1.24	64.84 $\pm$ 0.98	63.76 $\pm$ 1.87
SFT-noise	71.71 $\pm$ 8.20	43.00 $\pm$ 3.42	72.10 $\pm$ 0.98	50.63 $\pm$ 0.50	63.67 $\pm$ 0.12	66.00 $\pm$ 0.86	59.50 $\pm$ 4.01	62.88 $\pm$ 1.47
SFT-noisefull	77.06 $\pm$ 5.97	52.46 $\pm$ 5.80	74.72 $\pm$ 1.06	66.16 $\pm$ 6.06	64.20 $\pm$ 1.13	65.64 $\pm$ 0.81	67.22 $\pm$ 3.19	72.36 $\pm$ 0.70
SFT-noisetxt	75.64 $\pm$ 0.49	51.67 $\pm$ 1.22	76.35 $\pm$ 1.50	68.27 $\pm$ 6.29	64.78 $\pm$ 0.93	65.26 $\pm$ 1.23	67.49 $\pm$ 0.76	74.71 $\pm$ 1.30
SFT-simp	75.03 $\pm$ 4.71	43.57 $\pm$ 3.46	74.66 $\pm$ 2.67	51.16 $\pm$ 0.86	63.54 $\pm$ 1.87	64.49 $\pm$ 1.37	66.37 $\pm$ 3.40	64.79 $\pm$ 0.70
SFT-simf	53.38 $\pm$ 1.29	40.28 $\pm$ 1.88	71.66 $\pm$ 3.38	51.52 $\pm$ 1.24	63.70 $\pm$ 2.10	65.92 $\pm$ 0.86	59.42 $\pm$ 1.24	51.24 $\pm$ 1.35

Table 45: GraphLLM’s results in the transductive/poisoning setting against textual attacks. (ptb\_rate=0.80)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History
GraphGPT	63.54 $\pm$ 2.39	40.82 $\pm$ 11.87	84.52 $\pm$ 1.46	61.96 $\pm$ 3.44	42.13 $\pm$ 3.52	46.42 $\pm$ 2.04	73.18 $\pm$ 1.18
GraphGPT-noise	62.94 $\pm$ 1.39	41.01 $\pm$ 5.32	86.25 $\pm$ 2.95	62.19 $\pm$ 1.42	38.71 $\pm$ 1.55	46.83 $\pm$ 2.65	70.51 $\pm$ 0.98
LLaGA	78.96 $\pm$ 2.60	52.52 $\pm$ 1.31	72.16 $\pm$ 8.62	75.60 $\pm$ 3.10	36.33 $\pm$ 0.26	50.67 $\pm$ 6.62	75.57 $\pm$ 4.36
LLaGA-noise	79.05 $\pm$ 3.07	51.87 $\pm$ 10.89	78.33 $\pm$ 5.89	71.42 $\pm$ 2.64	36.82 $\pm$ 0.78	47.41 $\pm$ 3.08	71.54 $\pm$ 2.20
LLaGA-noisefull	77.66 $\pm$ 2.19	50.24 $\pm$ 5.85	80.16 $\pm$ 1.53	70.41 $\pm$ 7.15	42.64 $\pm$ 4.51	62.39 $\pm$ 2.86	75.22 $\pm$ 5.33
LLaGA-noisetxt	76.27 $\pm$ 4.31	51.96 $\pm$ 4.09	81.46 $\pm$ 2.41	73.94 $\pm$ 3.49	41.82 $\pm$ 2.11	63.95 $\pm$ 1.00	76.16 $\pm$ 3.77
SFT-neighbor	72.40 $\pm$ 4.62	46.44 $\pm$ 4.32	93.07 $\pm$ 0.55	79.33 $\pm$ 1.10	62.11 $\pm$ 2.91	59.12 $\pm$ 6.06	79.88 $\pm$ 3.15
SFT-noise	73.99 $\pm$ 3.44	47.89 $\pm$ 4.41	93.49 $\pm$ 0.26	80.04 $\pm$ 0.41	61.86 $\pm$ 3.34	60.59 $\pm$ 2.97	82.07 $\pm$ 0.86
SFT-noisefull	74.59 $\pm$ 1.40	47.33 $\pm$ 10.74	92.94 $\pm$ 0.43	79.11 $\pm$ 1.12	63.88 $\pm$ 0.22	63.26 $\pm$ 0.57	81.46 $\pm$ 1.58
SFT-noisetxt	72.73 $\pm$ 4.19	43.87 $\pm$ 4.94	93.07 $\pm$ 0.47	79.21 $\pm$ 1.17	63.35 $\pm$ 0.64	61.62 $\pm$ 1.54	80.24 $\pm$ 0.88
SFT-simp	76.64 $\pm$ 1.70	46.92 $\pm$ 6.90	93.04 $\pm$ 0.22	79.51 $\pm$ 1.76	55.40 $\pm$ 14.53	60.41 $\pm$ 2.40	79.38 $\pm$ 1.78

Table 46: Accuracy under the inductive/evasion setting against structural attack. (ptb\_rate=0.1, atk\_emb=BoW, def\_emb=RoBERTa)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Computer	ArXiv	Avg Rank
GCN	77.24 $\pm$ 2.39	64.99 $\pm$ 1.81	84.58 $\pm$ 0.38	45.55 $\pm$ 0.84	59.41 $\pm$ 0.81	62.89 $\pm$ 1.72	70.36 $\pm$ 0.50	66.13 $\pm$ 0.85	64.39 $\pm$ 0.03	52.21 $\pm$ 0.00	10.00
GAT	77.80 $\pm$ 0.97	67.35 $\pm$ 1.41	84.41 $\pm$ 0.66	58.89 $\pm$ 0.37	62.70 $\pm$ 0.78	60.87 $\pm$ 1.07	72.41 $\pm$ 1.17	66.73 $\pm$ 0.30	64.82 $\pm$ 0.59	60.07 $\pm$ 0.00	7.80
APPNP	81.80 $\pm$ 2.05	72.73 $\pm$ 1.44	89.78 $\pm$ 0.28	78.76 $\pm$ 1.08	64.15 $\pm$ 1.01	56.53 $\pm$ 0.59	80.84 $\pm$ 0.39	70.76 $\pm$ 0.44	64.35 $\pm$ 0.23	62.73 $\pm$ 0.00	5.10
GPRGNN	81.80 $\pm$ 2.32	72.15 $\pm$ 0.70	89.96 $\pm$ 0.20	78.29 $\pm$ 0.98	60.95 $\pm$ 0.38	59.97 $\pm$ 1.01	82.50 $\pm$ 0.32	71.98 $\pm$ 0.51	69.51 $\pm$ 0.67	68.59 $\pm$ 0.00	4.20
RobustGCN	76.75 $\pm$ 2.27	65.57 $\pm$ 1.07	83.72 $\pm$ 0.30	51.94 $\pm$ 1.23	64.79 $\pm$ 0.79	57.29 $\pm$ 1.36	69.21 $\pm$ 0.40	64.10 $\pm$ 0.36	60.71 $\pm$ 0.98	57.20 $\pm$ 0.00	10.00
GCORN	77.86 $\pm$ 1.05	66.46 $\pm$ 0.66	85.38 $\pm$ 0.08	46.55 $\pm$ 0.26	59.69 $\pm$ 1.00	66.91 $\pm$ 1.50	65.15 $\pm$ 0.69	62.61 $\pm$ 0.11	56.28 $\pm$ 0.23	49.74 $\pm$ 0.00	9.70
NoisyGCN	77.31 $\pm$ 1.74	65.78 $\pm$ 1.58	84.46 $\pm$ 0.47	46.65 $\pm$ 0.79	60.95 $\pm$ 1.02	63.26 $\pm$ 1.37	70.52 $\pm$ 0.47	66.29 $\pm$ 0.72	64.75 $\pm$ 0.21	51.40 $\pm$ 0.00	8.80
GRAND	81.73 $\pm$ 1.45	72.68 $\pm$ 0.83	87.04 $\pm$ 0.25	75.78 $\pm$ 1.08	65.29 $\pm$ 0.25	63.45 $\pm$ 1.51	79.41 $\pm$ 0.33	74.37 $\pm$ 0.07	71.06 $\pm$ 0.33	70.93 $\pm$ 0.00	3.70
SoftmedianGDC	81.80 $\pm$ 2.70	71.47 $\pm$ 1.55	87.82 $\pm$ 0.08	81.57 $\pm$ 0.75	62.18 $\pm$ 0.65	—	80.89 $\pm$ 0.39	—	—	—	5.00
EvenNet	80.75 $\pm$ 1.06	72.78 $\pm$ 1.22	88.44 $\pm$ 0.26	70.94 $\pm$ 0.60	<b>66.80 <math>\pm</math> 0.16</b>	62.01 $\pm$ 2.06	76.49 $\pm$ 0.12	69.84 $\pm$ 0.30	68.87 $\pm$ 0.20	59.48 $\pm$ 0.00	4.90
ElasticGNN	78.23 $\pm$ 1.38	65.46 $\pm$ 1.03	85.49 $\pm$ 0.27	66.61 $\pm$ 0.63	60.05 $\pm$ 3.30	56.66 $\pm$ 0.77	74.21 $\pm$ 0.59	70.20 $\pm$ 0.15	63.96 $\pm$ 0.58	60.48 $\pm$ 0.00	8.30
GNNGuard	83.64 $\pm$ 1.94	73.67 $\pm$ 0.22	90.26 $\pm$ 0.33	83.10 $\pm$ 0.31	63.93 $\pm$ 0.57	58.52 $\pm$ 0.42	<b>84.78 <math>\pm</math> 0.42</b>	69.12 $\pm$ 0.65	68.60 $\pm$ 0.44	<b>71.26 <math>\pm</math> 0.00</b>	3.40
RUNG	<b>84.75 <math>\pm</math> 1.15</b>	<b>74.09 <math>\pm</math> 0.70</b>	<b>90.64 <math>\pm</math> 0.19</b>	<b>83.38 <math>\pm</math> 0.59</b>	65.18 $\pm$ 1.08	58.50 $\pm$ 0.53	—	—	—	—	2.67

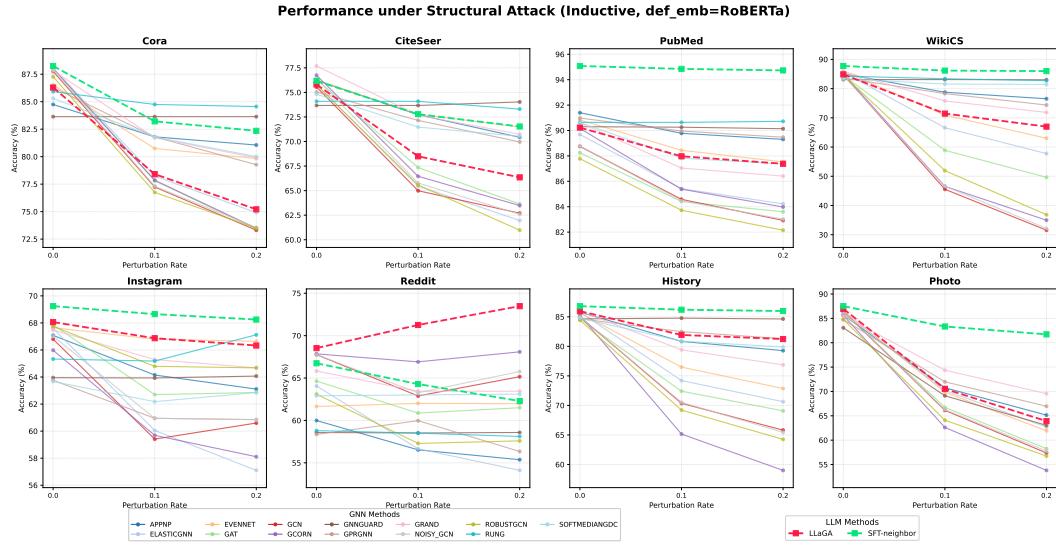


Figure 12: Performance under structural attacks (inductive setting). Accuracy trends on eight datasets as the perturbation rate increases. Each curve corresponds to a different GNN/RGNN-/GraphLLM method.

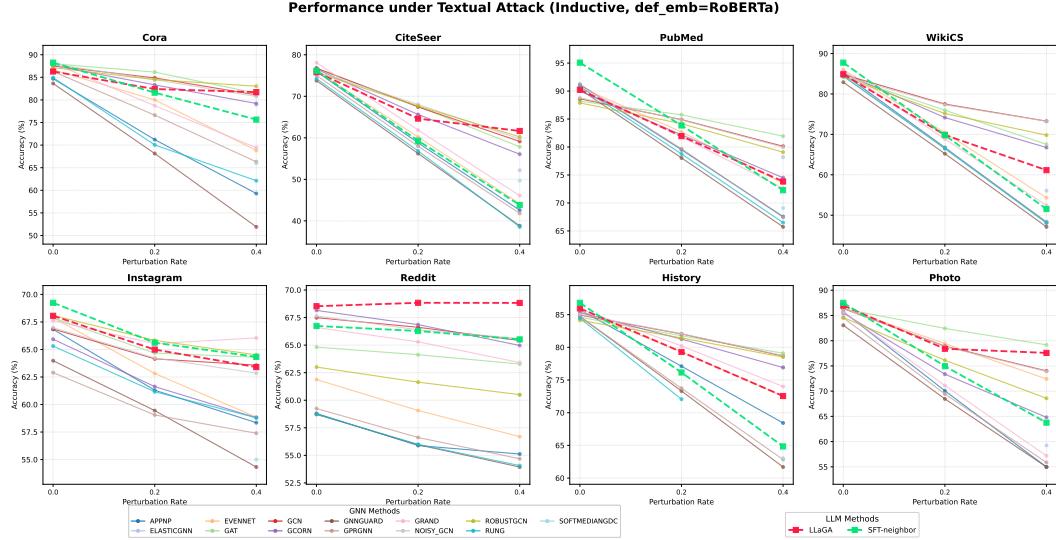


Figure 13: Performance under textual attacks (inductive setting). Accuracy trends on eight datasets as the perturbation rate increases. Each curve corresponds to a different GNN/RGNN/GraphLLM method.

Table 47: Accuracy under the inductive/evasion setting against textual attack. (ptb\_rate=0.2, def\_emb=RoBERTa)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo	Avg Rank
GCN	$84.87 \pm 0.40$	$67.45 \pm 0.63$	$84.90 \pm 0.30$	<b><math>77.50 \pm 0.45</math></b>	$64.15 \pm 0.45$	$66.62 \pm 0.67$	$82.05 \pm 0.28$	$78.92 \pm 0.19$	2.62
GAT	<b><math>86.16 \pm 1.83</math></b>	$67.55 \pm 1.34$	<b><math>85.76 \pm 0.74</math></b>	$75.98 \pm 2.19$	$64.67 \pm 1.06$	$64.13 \pm 0.20$	$81.76 \pm 0.93$	<b><math>82.44 \pm 0.33</math></b>	2.50
APPNP	$71.22 \pm 1.20$	$58.78 \pm 0.66$	$79.51 \pm 0.75$	$66.75 \pm 0.56$	$61.30 \pm 0.42$	$55.89 \pm 0.67$	$77.11 \pm 0.26$	$70.07 \pm 0.05$	8.62
GRNGUARD	$76.63 \pm 0.71$	$57.99 \pm 0.46$	$79.68 \pm 0.60$	$66.62 \pm 0.67$	$59.05 \pm 2.20$	$56.62 \pm 0.84$	$73.74 \pm 1.08$	$69.45 \pm 0.69$	8.88
RobustGCN	$84.50 \pm 0.84$	$67.66 \pm 0.52$	$84.17 \pm 0.57$	$75.17 \pm 1.68$	<b><math>65.83 \pm 0.16</math></b>	$61.64 \pm 0.71$	$81.44 \pm 0.66$	$76.13 \pm 0.83$	3.62
GCORN	$83.33 \pm 0.53$	$65.62 \pm 0.37$	$82.18 \pm 0.05$	$74.16 \pm 0.55$	$61.63 \pm 0.54$	<b><math>66.86 \pm 0.49</math></b>	$81.26 \pm 0.66$	$73.37 \pm 0.16$	5.00
NoisyGCN	$84.50 \pm 0.40$	<b><math>68.03 \pm 1.48</math></b>	$84.84 \pm 0.48$	$77.33 \pm 0.49$	$64.23 \pm 0.94$	$66.39 \pm 1.07$	<b><math>82.08 \pm 0.38</math></b>	$78.86 \pm 0.12$	2.62
GRAND	$78.72 \pm 0.38$	$61.86 \pm 1.10$	$81.83 \pm 0.53$	$68.86 \pm 0.82$	$65.53 \pm 0.67$	$65.30 \pm 1.00$	$80.21 \pm 0.43$	$71.11 \pm 0.15$	5.75
EvenNet	$80.01 \pm 1.48$	$60.19 \pm 1.12$	$82.69 \pm 0.78$	$70.41 \pm 1.53$	$62.82 \pm 1.36$	$59.07 \pm 0.90$	$79.32 \pm 0.63$	<b><math>79.44 \pm 0.21</math></b>	5.75
GNNGuard	$68.14 \pm 1.32$	$56.22 \pm 0.07$	$78.05 \pm 0.97$	$65.23 \pm 0.41$	$59.45 \pm 0.77$	$55.96 \pm 0.29$	$73.32 \pm 0.50$	$68.47 \pm 0.25$	10.50
RUNG	$70.05 \pm 1.26$	$56.84 \pm 0.53$	$78.76 \pm 0.38$	$66.47 \pm 0.50$	$61.16 \pm 0.57$	$56.00 \pm 0.77$	—	—	9.67

Table 48: GraphLLM’s results under the inductive/evasion setting against structural attacks. (ptb\_rate=0.10)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo
LLaGA	$78.41 \pm 0.64$	$68.49 \pm 1.50$	$87.97 \pm 1.28$	$71.46 \pm 3.37$	$66.87 \pm 1.03$	$71.26 \pm 5.87$	$81.93 \pm 1.56$	$70.53 \pm 2.44$
SFT-neighbor	$83.21 \pm 1.21$	$72.78 \pm 2.28$	$94.85 \pm 0.49$	$86.17 \pm 0.90$	$68.65 \pm 0.28$	$64.29 \pm 2.96$	$86.21 \pm 0.64$	$83.33 \pm 0.60$

Table 49: GraphLLM’s results under the inductive/evasion setting against textual attacks. (ptb\_rate=0.20)

Method	Cora	CiteSeer	PubMed	WikiCS	Instagram	Reddit	History	Photo
LLaGA	$82.41 \pm 1.86$	$64.63 \pm 1.81$	$81.94 \pm 1.12$	$69.86 \pm 2.27$	$64.99 \pm 1.35$	$68.84 \pm 1.97$	$79.29 \pm 1.14$	$78.40 \pm 0.64$
SFT-neighbor	$81.61 \pm 0.21$	$59.30 \pm 2.28$	$83.85 \pm 1.25$	$69.80 \pm 0.64$	$65.62 \pm 1.10$	$66.28 \pm 1.35$	$76.15 \pm 1.15$	$74.94 \pm 0.51$

## K ADDITIONAL BASELINE EVALUATION

To further validate the persistence of the text-structure robustness trade-off, we evaluate three recent methods: GPRGNN-AT Gosch et al. (2023) (with PR-BCD training), GOOD-AT Li et al. (2024), and GPR-GAE Lee & Park (2025). Tables 50 and 51 present results under inductive structural and textual attacks, respectively.

Table 50: Performance under inductive structural attacks (PGD 20%). Format: Clean/Attacked (%).

Method	Cora	CiteSeer	PubMed	WikiCS
GCN	87.76/73.31	76.07/62.70	88.76/82.92	84.86/31.54
GPRGNN	86.29/79.27	74.19/69.96	91.05/89.47	84.38/74.38
GPRGNN-AT	86.10/81.92	74.03/71.32	90.61/90.49	83.11/81.43
GOOD-AT	88.44/73.68	76.18/63.27	88.83/82.99	84.85/32.09
GPR-GAE	87.82/79.03	76.44/ <b>74.71</b>	88.48/88.39	84.19/41.73
SFT-Auto	83.27/ <b>82.59</b>	74.56/74.24	94.74/ <b>92.29</b>	86.40/ <b>84.05</b>

Table 51: Performance under inductive textual attacks (GPT 40%). Format: Clean/Attacked (%).

Method	Cora	CiteSeer	PubMed	WikiCS
GCN	87.76/81.12	76.07/59.14	88.76/80.13	84.86/73.27
GPRGNN	86.29/66.30	74.19/41.80	91.05/67.43	84.38/48.40
GPRGNN-AT	86.10/65.68	74.03/39.71	90.61/66.78	83.11/47.20
GOOD-AT	88.44/ <b>83.03</b>	76.18/59.93	88.83/ <b>80.35</b>	84.85/75.35
GPR-GAE	87.82/67.10	76.44/40.07	88.48/70.86	84.19/66.38
SFT-Auto	83.27/76.75	74.56/ <b>65.36</b>	94.74/77.86	86.40/ <b>77.83</b>

**Key Observations:** The results confirm that the text-structure trade-off persists in adversarial training methods. GPR-GAE and GPRGNN-AT significantly improve structural robustness but *degrade* textual robustness. Conversely, GOOD-AT excels under textual attacks (83.03% on Cora) but collapses under structural attacks (32.09% on WikiCS). This validates our finding that existing defenses exhibit inherent trade-offs, while SFT-Auto achieves balanced robustness across both modalities.