Learning on Incomplete Graphs: Benchmarking GNN Robustness to Sparsification

Rakibul Hassan and Shaikh Arifuzzaman
Department of Computer Science
University of Nevada, Las Vegas
Las Vegas, NV, USA
Emails: hassar2@unlv.nevada.edu, shaikh.arifuzzaman@unlv.edu

***Abstract*—Graph Neural Networks (GNNs) have achieved strong performance across a wide range of graph learning applications. However, real-world graphs are often incomplete, noisy, or intentionally sparsified to satisfy computational and memory constraints, raising important questions about model robustness under structural degradation. In this paper, we present a systematic empirical benchmark of GNN robustness under graph sparsification. We evaluate three representative architectures: Graph Convolutional Networks (GCN), GraphSAGE, and Graph Attention Networks (GAT) using seven sparsification strategies spanning random, structural, feature-aware, and spectral paradigms. Experiments on the Cora, CiteSeer, and PubMed datasets comprise 693 unique model–dataset–sparsification configurations across multiple sparsity levels.**

The results reveal clear architecture and method-dependent robustness trends. Feature-aware approaches, particularly Top-k Similarity Pruning, consistently achieve the strongest robustness, preserving approximately 95% of baseline predictive performance on average. Spectral Sparsification emerges as the most effective structure-preserving alternative, while Hub Removal and k-Core Decomposition exhibit larger performance degradation. Robustness is also strongly dataset-dependent, with PubMed demonstrating substantially greater tolerance to edge reduction than Cora and CiteSeer. Across architectures, GCN and GAT generally maintain higher performance under graph reduction, whereas GraphSAGE is more sensitive to structural perturbations. These findings provide practical insights for selecting sparsification strategies and GNN architectures in noisy, incomplete, and resource-constrained graph learning environments.

***Index Terms*—Graph Neural Networks, GCN, GraphSAGE, Robustness, Sparsification**

I. Introduction

Graph-structured data naturally arise in numerous real-world applications, including citation [1] and knowledge networks [2], social media analysis [3], recommender systems [4], cybersecurity monitoring [5], biological interaction networks [6], and healthcare analytics [7]. Learning meaningful representations from such relational data is critical for tasks such as node classification, link prediction, anomaly detection, and recommendation. In recent years, Graph Neural Networks (GNNs) have emerged as a dominant paradigm for graph representation learning, achieving state-of-the-art performance across a wide range of graph learning tasks [8]–[10]. Representative architectures such as Graph Convolutional Networks (GCN) [8], GraphSAGE [9], and Graph Attention Networks (GAT) [10] learn expressive node embeddings through iterative neighborhood aggregation, enabling effective modeling of both structural and attribute information.

Despite the success, deploying GNNs in real-world applications remains challenging due to the scale, density, and imperfect nature of practical graphs [11], [12]. Modern graph datasets often contain millions of nodes and billions of edges [13], leading to substantial computational and memory overhead during message passing. Furthermore, real-world graphs are rarely fully reliable, as edges may be missing due to incomplete observations, removed for privacy preservation, corrupted by noise, or intentionally reduced to satisfy resource constraints [11], [14], [15]. Such structural degradation is common in social networks with partial user interactions, communication graphs collected under packet loss, citation networks with evolving relationships, and biological networks with incomplete experimental evidence. In these settings, understanding how GNNs behave under structural sparsification is essential for reliable deployment.

Graph sparsification seeks to reduce graph complexity while preserving important structural properties [16]. Classical sparsification methods have been extensively studied in graph algorithms and scientific computing, where the primary objective is to reduce graph complexity while preserving important structural properties such as connectivity, graph cuts, and spectral characteristics [16], [17]. However, these methods were not originally designed for graph representation learning, where preserving task-relevant neighborhoods and feature propagation pathways is often more important than preserving purely graph-theoretic properties. Consequently, removing edges without considering their effect on message passing can significantly alter learned representations and predictive performance. Moreover, different GNN architectures may respond differently to sparsification due to their distinct aggregation mechanisms, sampling strategies, and attention-based weighting schemes.

Although recent studies have investigated graph sparsification, edge dropout, and graph sampling techniques for improving GNN scalability and training efficiency [14], [18], [19], a comprehensive evaluation of how different sparsification paradigms affect robustness across multiple GNN architectures and benchmark datasets remains largely unexplored.

In this paper, we conduct a comprehensive empirical study of graph sparsification for Graph Neural Networks (GNNs).

We evaluate **seven** representative edge reduction strategies spanning four complementary sparsification paradigms: random, structural, feature-based, and spectral. The evaluated methods include random edge pruning, degree-based hub pruning, local degree-based top-k pruning, k-core decomposition, similarity based thresholding, top-k similarity pruning, and spectral sparsification. These methods are evaluated across three widely used benchmark datasets (Cora, CiteSeer, and PubMed) and **three** representative GNN architectures: GCN, GraphSAGE, and GAT. Overall, our study comprises **693 unique** experiment configurations across multiple sparsity levels, enabling a systematic analysis of robustness, accuracy retention, and architecture-dependent behavior under progressive structural degradation.

The main contributions of this paper are summarized as follows:

- To the best of our knowledge, this paper provides one of the **most comprehensive** empirical evaluations of graph sparsification robustness for Graph Neural Networks, spanning **seven** sparsification strategies with **eleven** sparsity levels, **three** benchmark datasets, and **three** representative GNN architectures.
- We systematically evaluate four sparsification **paradigms** (random, structural, feature-based, and spectral) to characterize their impact on predictive performance under progressive graph reduction.
- We employ quantitative robustness metrics based on **accuracy retention** to compare sparsification strategies across datasets and models.
- We demonstrate that carefully designed sparsification can preserve **over** 90% **of baseline** accuracy while substantially reducing graph connectivity.
- We provide practical insights for deploying robust and efficient graph learning systems in noisy, incomplete, and resource-constrained environments.

The remainder of this paper is organized as follows. Section II reviews existing work on graph sparsification and Graph Neural Networks. Section III introduces the foundational concepts underlying GNNs and graph sparsification. Section IV presents the proposed benchmarking framework, sparsification taxonomy, datasets, experimental setup, and evaluation matrices. Section V reports the experimental results and provides a comprehensive comparative analysis of sparsification robustness across datasets and architectures. Finally, Section VI contains the conclusions and outlines directions for future research.

II. Related Work

Graph Neural Networks (GNNs) employ different neighborhood aggregation mechanisms that can lead to distinct behavior under graph perturbations. Graph Convolutional Networks (GCNs) aggregate normalized neighborhood information through spectral-inspired convolutions [8], GraphSAGE performs inductive learning through sampled neighborhood aggregation [9], while Graph Attention Networks (GATs) dynamically weight neighboring nodes using learned attention coefficients [10]. These architectural differences have motivated extensive research into scalable and robust graph learning methods [3], [11].

To address the computational challenges of large-scale graphs, several studies have explored graph reduction and sampling techniques. DropEdge [14] introduces random edge removal during training to alleviate over-smoothing in deep GNNs. Cluster-GCN [18] improves scalability through graph partitioning, while GraphSAINT [19] employs subgraph sampling to reduce training costs. These approaches demonstrate that reducing graph connectivity can significantly improve computational efficiency while maintaining competitive predictive performance.

Recent surveys have identified noisy, incomplete, and structurally perturbed graphs as important challenges for real-world GNN deployment [15]. However, existing studies primarily focus on a specific reduction technique, training strategy, or model architecture. A systematic understanding of how fundamentally different sparsification paradigms affect multiple GNN architectures remains limited. In contrast, our work provides a unified benchmarking framework that evaluates seven sparsification strategies spanning random, structural, feature-based, and spectral paradigms across GCN, GraphSAGE, and GAT on three widely used citation-network benchmarks.

III. Background

This section introduces the theoretical foundations of graph neural networks and graph sparsification that underpin our study.

A. Graph Representation and Message Passing

A graph is represented as $G = (V, E, X, Y)$, where V denotes the set of nodes, E denotes the set of edges, $X \in \mathbb{R}^{|V|\times d}$ represents node feature vectors of dimension d, and Y represents class labels. The connection among the nodes is represented using an adjacency matrix $A \in \mathbb{R}^{|V|\times|V|}$, where $A_{ij} = 1$ indicates an edge between nodes i and j.

Graph Neural Networks (GNNs) learn node representations through iterative neighborhood aggregation, commonly referred to as message passing [11], [20]. At layer l, the representation of node v is updated by aggregating information from its neighbors:

$$h_v^{(l+1)} = \text{Update}\left(h_v^{(l)}, \text{AGG}\left(\{h_u^{(l)} : u \in \mathcal{N}(v)\}\right)\right), \quad (1)$$

where $\mathcal{N}(v)$ denotes the neighborhood of node v, and $h_v^{(l)}$ represents its embedding at layer l.

B. Graph Neural Network Architectures

1) Graph Convolutional Networks (GCN): Graph Convolutional Networks (GCNs) introduced by [8] perform neighborhood aggregation using symmetric normalization of the adjacency matrix. A GCN layer is defined as:

$$H^{(l+1)} = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right), \quad (2)$$

where $\hat{A} = A + I$ adds self-loops, $\hat{D}$ is the degree matrix, $W^{(l)}$ is a learnable weight matrix, and $\sigma(\cdot)$ denotes a nonlinear activation function. GCNs effectively capture local homophilic patterns but rely heavily on graph connectivity.

2) GraphSAGE: GraphSAGE extends neighborhood aggregation to inductive settings by sampling fixed-size neighborhoods and applying learnable aggregation functions [9]. The GraphSAGE update can be expressed as:

$$h_v^{(l+1)} = \sigma\left(W^{(l)} \cdot \text{CONCAT}\left(h_v^{(l)}, \text{AGG}\left(\{h_u^{(l)} : u \in \mathcal{N}(v)\}\right)\right)\right) \quad (3)$$

This sampling-based design improves scalability for large graphs but may increase sensitivity to neighborhood perturbations.

3) Graph Attention Networks (GAT): Graph Attention Networks (GATs) use self-attention to assign adaptive importance weights to neighboring nodes during aggregation [10]. For node v, attention coefficients are computed as:

$$\alpha_{vu} = \frac{\exp\left(\text{LeakyReLU}(a^T[Wh_v||Wh_u])\right)}{\sum_{k\in\mathcal{N}(v)}\exp\left(\text{LeakyReLU}(a^T[Wh_v||Wh_k])\right)} \quad (4)$$

where α_{vu} denotes the importance of neighbor u. The adaptive weighting mechanism allows GAT to emphasize informative neighbors and suppress noisy connections.

C. Sparsification Strategies

We investigate seven sparsification strategies spanning random, structural, feature-driven, and spectral paradigms.

1) Random Edge Pruning: Random pruning removes edges uniformly at random, similar in spirit to DropEdge regularization [14]. Although computationally simple, it does not explicitly preserve graph topology or feature similarity.

2) Degree-Based Hub Pruning: Hub pruning removes edges incident to the network's highest-degree nodes. This strategy is motivated by the mitigation of global structural bottlenecks and over-squashing in message passing [21], allowing the network to evaluate the impact of highly connected hubs on information flow.

3) Local (Top-k) Degree-Based Pruning: Local degree-based filtering selectively prunes edges by evaluating neighbor importance from the perspective of each individual node [22]. For each target node v, its incoming or outgoing edges are sorted in descending order based on the structural degree of the connected neighbor $u \in \mathcal{N}(v)$. The algorithm retains only the top $(1-p)$ fraction of these edges (where p is the removal ratio), guaranteeing that every active node preserves its most structurally significant local connections while adaptively stripping away minor peripheral noise [23].

.

4) k-Core Decomposition: This topological filter strips away peripheral shells to isolate the maximal induced subgraph where all remaining vertices maintain an internal degree of at least k [24]. Restricting message passing to the k-core ensures that node embeddings are generated purely from densely connected, structurally resilient subgraphs, entirely shielded from low-degree boundary noise.

5) Similarity based Thresholding/Pruning: Similarity-based thresholding/pruning transitions from purely structural filtering to feature-driven topology refinement. This method evaluates existing edges by computing the cosine similarity between the feature vectors of connected node pairs:

$$\text{sim}(u, v) = \frac{\mathbf{x}_u \cdot \mathbf{x}_v}{|\mathbf{x}_u||\mathbf{x}_v|}. \quad (5)$$

Edges whose pairwise similarity falls below a globally determined quantile threshold τ are systematically pruned [25].

6) Top-k Similarity Pruning: Instead of enforcing a global similarity threshold, this strategy operates from a node-centric perspective to enable adaptive local feature preservation [26]. For each node v, the cosine similarity is computed between its feature vector $\mathbf{x}_v$ and the vectors of all its immediate neighbors $u \in \mathcal{N}(v)$. The local neighborhood is then ranked in descending order based on these similarity scores, retaining only the top-k highest-ranked connections (or a specific top percentile) while dropping remaining edges.

7) Spectral Sparsification: Spectral sparsification reduces the number of edges while preserving the spectral properties of the original graph. It selects edges based on their spectral importance so that the graph Laplacian of the sparsified graph remains a close approximation of the original [16]. Unlike local pruning methods that rely on node degree or feature similarity, spectral sparsification preserves global connectivity and diffusion characteristics, helping maintain effective message passing and predictive performance even under substantial edge reduction.

IV. Methodology

This section presents the proposed sparsification-aware benchmarking framework developed to systematically evaluate the robustness of graph neural networks under progressive structural degradation.

A. Proposed Benchmarking Framework

Figure 1 provides an overview of the proposed experimental pipeline. The framework consists of six major components: (1) benchmark graph selection, (2) sparsification strategy selection, (3) graph reduction, (4) GNN training, (5) experiments, and (6) robustness evaluation. Given an input graph $G = (V, E, X, Y)$, the proposed framework first selects a graph sparsification strategy from a diverse taxonomy of reduction paradigms. The selected method is then applied to generate a sparsified graph $G' = (V, E', X, Y)$ where $E' \subseteq E$.

The reduced graph is subsequently used to train multiple graph neural network architectures under identical experimental settings. Finally, model performance is evaluated across progressively increasing sparsity levels to quantify robustness, accuracy preservation, and architecture-dependent degradation behavior.

B. Sparsification Taxonomy

To enable a comprehensive evaluation of structural robustness, we implement seven sparsification strategies spanning four fundamentally different graph reduction paradigms:

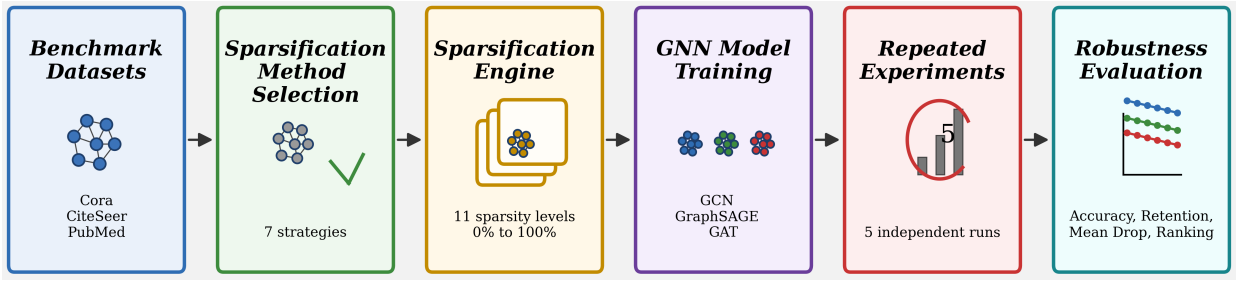

Fig. 1. Overall workflow of the proposed sparsification-aware benchmarking framework. The framework generates multiple structurally perturbed graph instances and evaluates architecture-level robustness across progressive sparsity levels.

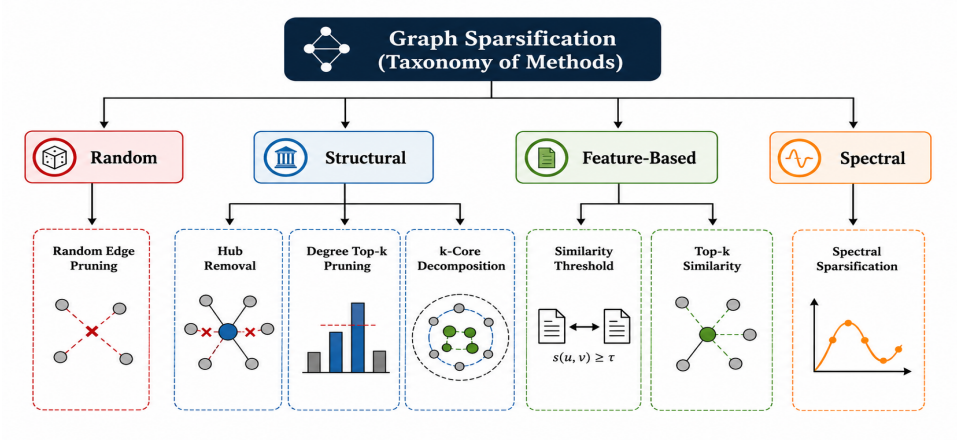

Fig. 2. Taxonomy of the graph sparsification strategies implemented in the proposed benchmarking framework. The evaluated methods are grouped into four categories: **Random**, **Structural** pruning based on node degree and graph topology (hub removal, degree top-k, and k-core decomposition), **Feature-Based** pruning using node similarity (thresholding and top-k similarity), and **Spectral** sparsification for preserving global Laplacian structure. This taxonomy enables systematic evaluation of topology-aware and feature-aware graph reduction strategies for robust GNN learning.

TABLE I
BENCHMARK DATASETS USED IN THIS STUDY.

Dataset	Nodes	Edges	Features	Classes
Cora	2708	10556	1433	7
CiteSeer	3327	9104	3703	6
PubMed	19717	88648	500	3

Random, Structural, Feature-Driven, and Spectral. Figure 2 illustrates the taxonomy of the implemented sparsification strategies. Together, these methods capture diverse perspectives on graph reduction, ranging from topology-preserving sparsification to feature-aware edge selection. This taxonomy enables a systematic comparison of how different sparsification paradigms affect information propagation and predictive performance across GNN architectures.

C. Benchmark Datasets

We evaluate the proposed framework on three widely benchmark datasets from the Planetoid collection: Cora, CiteSeer, and PubMed. In these datasets, nodes represent scientific publications, edges represent citation relationships, and node features correspond to bag-of-words representations. Table I summarizes the characteristics of the benchmark datasets.

D. Sparsification Implementation

For ratio-based sparsification methods, the reduction parameter is varied from 0.0 to 1.0 with step size 0.1, producing eleven sparsity levels ranging from the original graph to extreme edge reduction. For k-based methods, k is varied from 1 to 10 to control neighborhood preservation. For feature-driven methods, cosine similarity between node features is used to preserve semantically similar neighborhoods. For structural methods, node degree statistics and connectivity constraints guide edge selection. Spectral sparsification preserves edges that maintain global Laplacian structure.

E. GNN Model Configuration

To evaluate architecture-dependent robustness, we employ three representative graph neural network architectures: GCN, GraphSAGE, and GAT. All models use a two-layer architecture with 16 hidden units. GCN and GraphSAGE employ ReLU activation and a dropout rate of 0.5, while GAT uses ELU activation, 8 attention heads in the first layer, and a dropout rate of 0.6.

To ensure fair comparison, identical train, validation, and test masks are used across all sparsification settings. All models are trained using the Adam optimizer with a learning rate of 0.01 and weight decay of 5×10^{-4} for 200 epochs. The output layer size is determined by the number of classes in each dataset. To account for stochastic variation, each experiment is repeated five times using different random seeds, and the reported results correspond to the mean classification accuracy. All experiments were implemented using Python, PyTorch, and PyTorch Geometric. Training and evaluation

were performed on a workstation equipped with an NVIDIA RTX 3060 GPU.

F. Evaluation Metrics

Let $G = (V, E, X, Y)$ denote the original graph and $G' = (V, E', X, Y)$ denote the sparsified graph. We define edge sparsity, or edge reduction ratio, as:

$$S(G') = \left(1 - \frac{|E'|}{|E|}\right) \times 100. \tag{6}$$

Thus, $S = 0\%$ corresponds to the original graph, whereas $S = 100\%$ corresponds to complete edge removal.

In addition to raw accuracy, we report *accuracy retention* (AR), which measures the percentage of baseline predictive performance preserved after sparsification:

$$AR(s) = \frac{Acc(s)}{Acc(0)} \times 100. \tag{7}$$

where $Acc(0)$ is the accuracy on the original graph and $Acc(s)$ is the accuracy at edge sparsity level s. An AR value close to 100% indicates that the sparsified graph preserves nearly all of the original predictive performance. This normalized measure is useful because the three datasets have different baseline accuracies. In addition, we report High-Sparsity Retention (retention at sparsity levels above 70%) and Mean Accuracy Drop relative to the original graph.

V. Results and Analysis

This section presents the empirical evaluation of the proposed graph sparsification benchmarking framework. We analyze how predictive performance evolves under progressive graph reduction, compare the robustness of different sparsification strategies, and investigate the influence of dataset characteristics and GNN architectures on sparsification sensitivity.

A. Accuracy Trends Under Progressive Sparsification

To examine how graph sparsification affects predictive performance, Figure 3 presents the accuracy trajectories of GCN, GraphSAGE, and GAT across all seven sparsification strategies and three benchmark datasets. Each subplot reports the mean classification accuracy over five independent runs as the sparsification control parameter is progressively varied. Depending on the method, this parameter corresponds to an edge-removal ratio, neighborhood size, core number, similarity threshold, or spectral retention ratio.

We observe several important patterns here. First, predictive performance generally deteriorates as sparsification becomes more aggressive, although the degradation rate varies considerably across methods and architectures. Structural pruning approaches such as Hub pruning and k-Core Decomposition typically produce the largest performance losses, particularly on Cora and CiteSeer, indicating greater sensitivity to the removal of structurally important edges. In contrast, feature-aware approaches, especially Similarity based thresholding and top-k similarity pruning, maintain substantially more stable performance and occasionally improve accuracy under mild sparsification, suggesting a denoising effect. Spectral Sparsification consistently preserves strong predictive performance across all datasets, with PubMed exhibiting the highest robustness. Across architectures, GCN and GAT generally show smoother degradation trends, whereas GraphSAGE is more sensitive to aggressive structural perturbations.

TABLE II
Overall robustness summary of sparsification methods. Higher Accuracy Retention and High-Sparsity Retention indicate stronger robustness, while lower Mean Accuracy Drop is preferred.

Method	Accuracy Retention (%)	High-Sparsity Retention (%)	Mean Accuracy Drop
Top-k Similarity Pruning	94.63	83.33	4.11
Spectral Sparsification	91.49	82.17	6.37
Random Edge Pruning	90.42	81.97	7.16
Similarity Thresholding	89.83	81.17	7.61
Local Degree-Based Pruning	87.59	82.53	9.30
Degree-Based Hub pruning	84.07	81.45	11.94
k-Core Decomposition	82.69	76.46	12.94

Figure 4 shows the accuracy vs sparsity curves for each dataset and GNN architecture. Across all datasets, accuracy generally decreases as more edges are removed, but the degradation pattern differs substantially across sparsification methods. Random edge removal provides a useful lower-complexity baseline, but structure-aware and feature-aware approaches often preserve accuracy more effectively. This behavior is consistent with prior sparsification studies showing that smaller graphs can maintain useful predictive information when important graph structure is preserved. Table II summarizes Accuracy Retention, High-Sparsity Retention, and Mean Accuracy Drop across all methods.

Figure 5 reveals strong dataset dependence. PubMed is the most robust dataset, with an average accuracy retention of 94.37% across all experiments. CiteSeer follows with 88.02%, while Cora has the lowest average retention at 83.64%. This suggests that PubMed contains stronger feature or label redundancy, allowing GNNs to retain high accuracy even after aggressive edge reduction.

Figure 6 compares average accuracy retention across all methods. Top-k similarity pruning achieves the highest overall retention, with a mean retention of 94.63%. Spectral sparsification is the second-best method, with 91.49% mean retention. Random edge pruning and similarity based thresholding also remain competitive, while hub pruning and k-core decomposition show stronger degradation on average.

The superiority of Top-k similarity pruning indicates that preserving a small number of semantically relevant neighbors can be more beneficial than global thresholding or purely degree-driven pruning. This is especially useful for citation networks, where textual feature similarity often correlates with class membership. Spectral sparsification also performs strongly because it preserves global structural properties of the graph, making it less disruptive to message passing.

B. High-Sparsity Behavior

To evaluate robustness under aggressive graph compression, Figure 7 reports mean accuracy retention for runs with at least 70% edge reduction. Top-k Similarity Pruning achieves the

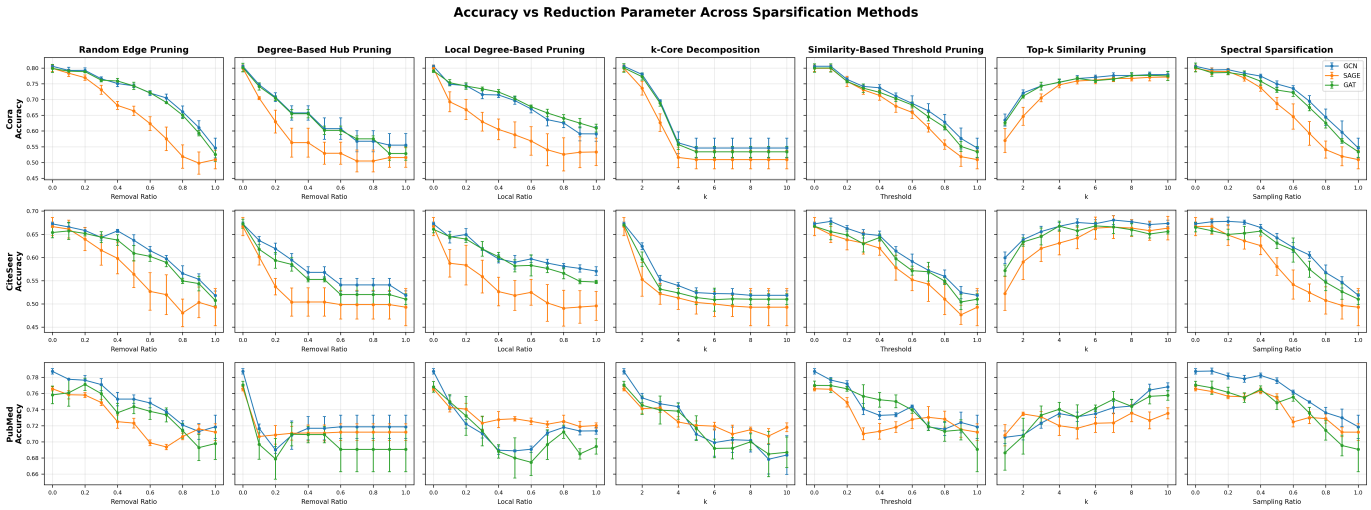

Fig. 3. Classification accuracy as a function of the sparsification control parameter for seven sparsification strategies across Cora, CiteSeer, and PubMed. Rows correspond to sparsification methods, columns correspond to datasets, and each subplot compares GCN, GraphSAGE, and GAT. Results are averaged over five independent runs.

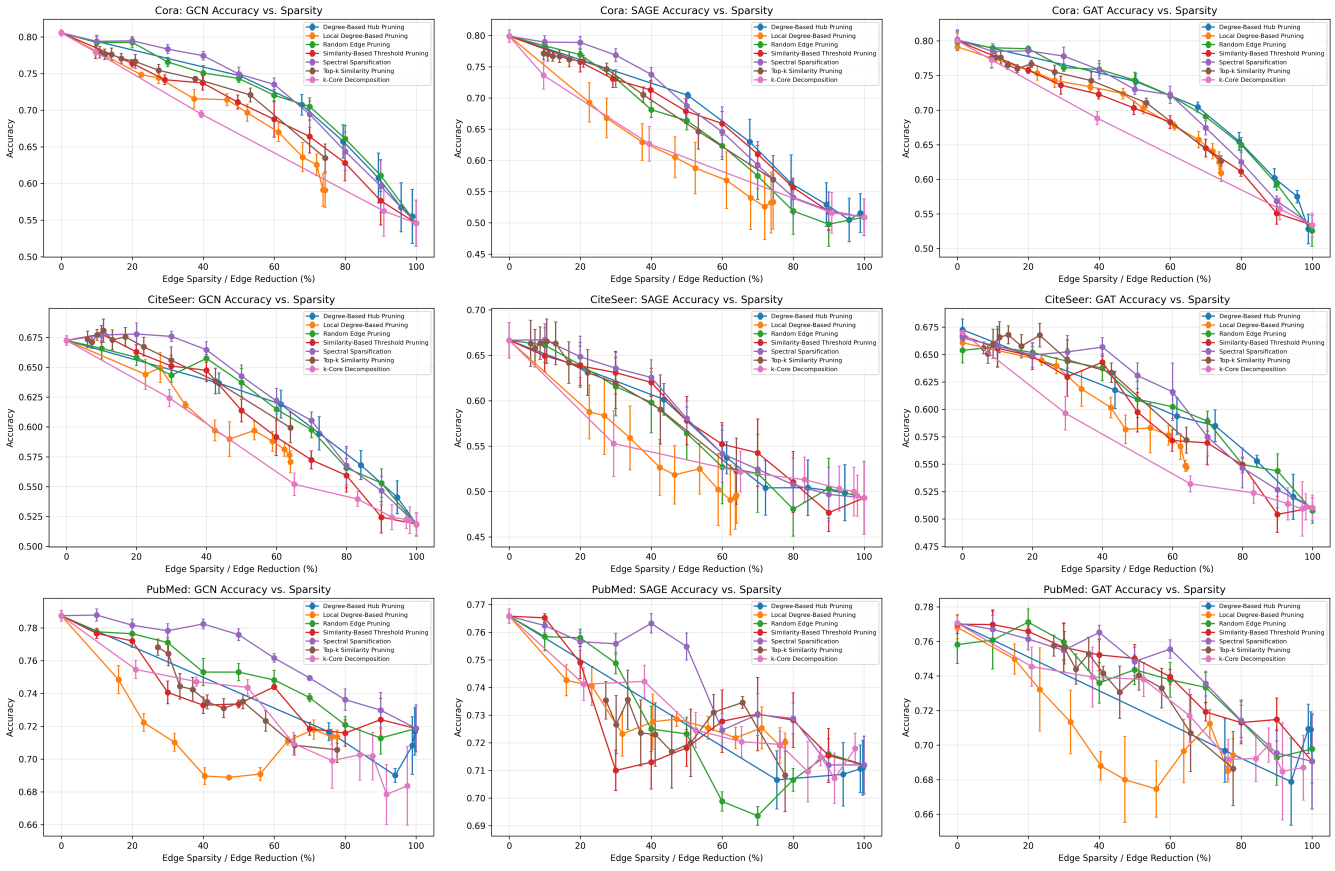

Fig. 4. Accuracy versus edge sparsity for GCN, GraphSAGE, and GAT across Cora, CiteSeer, and PubMed. Each curve corresponds to one sparsification method.

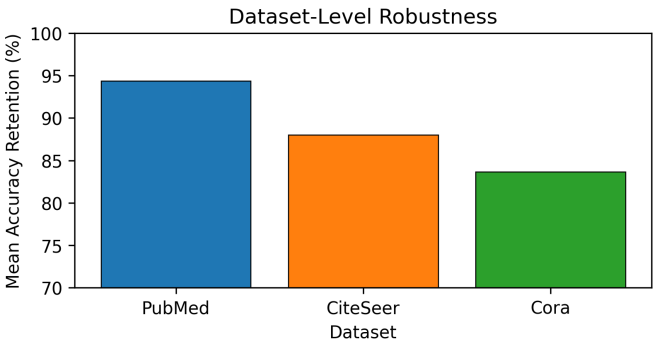

Fig. 5. Dataset-level robustness measured by mean accuracy retention across all sparsification methods and GNN architectures.

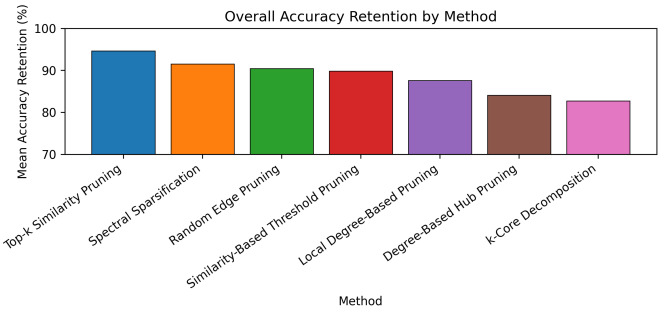

Fig. 6. Overall accuracy retention by sparsification method, averaged over datasets and GNN architectures. Higher values indicate stronger preservation of baseline predictive performance.

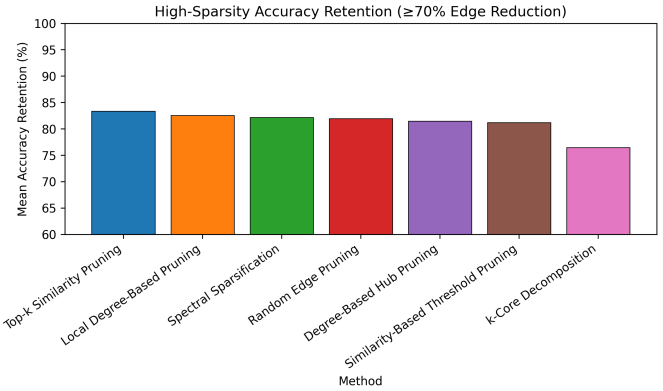

Fig. 7. Accuracy retention under high sparsity, considering only experiments with at least 70% edge reduction.

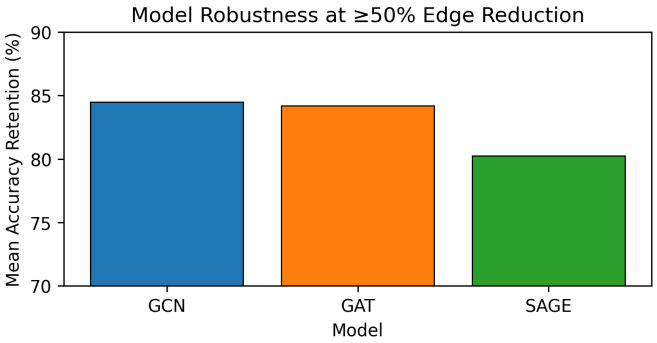

Fig. 8. Model-level robustness under moderate-to-high sparsity. Results are averaged over all datasets and methods with at least 50% edge reduction.

strongest performance, retaining 83.33% of baseline accuracy under high sparsity. Local Degree-Based Pruning, Spectral Sparsification, Random Edge Pruning, and Degree-Based Hub Pruning remain relatively competitive, whereas k-Core Decomposition exhibits the largest performance degradation in this regime.

C. Architecture-Level Sensitivity

Figure 8 compares model robustness under moderate-to-high sparsity, defined as at least 50% edge reduction. GCN achieves the highest average retention at 84.47%, closely followed by GAT at 84.21%. GraphSAGE shows lower retention, with 80.26% on average.

These results indicate that sparsification affects aggregation mechanisms differently. GCN benefits from normalized neighborhood aggregation and remains stable when the retained graph preserves global structure. GAT remains competitive because attention can adaptively reweight surviving neighbors. GraphSAGE is generally more sensitive to structural perturbations, suggesting that neighborhood aggregation becomes less reliable when informative connections are removed.

D. Model–Method Interaction

Figure 9 summarizes the interaction between GNN architectures and sparsification methods under moderate-to-high sparsity conditions. Each cell reports the mean accuracy retention aggregated across all datasets and sparsity levels within this regime. The heatmap largely reinforces the trends observed in previous analyses. Top-k Similarity Pruning consistently achieves the highest retention across all architectures, whereas k-Core Decomposition exhibits the largest degradation. GCN and GAT remain generally more robust than GraphSAGE, highlighting the importance of both sparsification strategy and aggregation mechanism in determining robustness under graph reduction.

E. Findings

Our experimental results demonstrate that graph sparsification can substantially reduce graph connectivity while preserving predictive performance when informative edges are retained. Feature-aware approaches, particularly Top-k Similarity Pruning, consistently achieve the strongest robustness across datasets and architectures, while Spectral Sparsification emerges as the most effective structure-preserving alternative. Robustness is strongly influenced by the characteristics of the dataset, and PubMed exhibits considerably greater tolerance to edge reduction than Cora and CiteSeer. The results also reveal clear architecture-dependent behavior, as GCN and GAT generally maintain higher performance under graph reduction, whereas GraphSAGE is more sensitive to structural perturbations. Overall, these findings suggest that effective sparsification should preserve both structural connectivity and

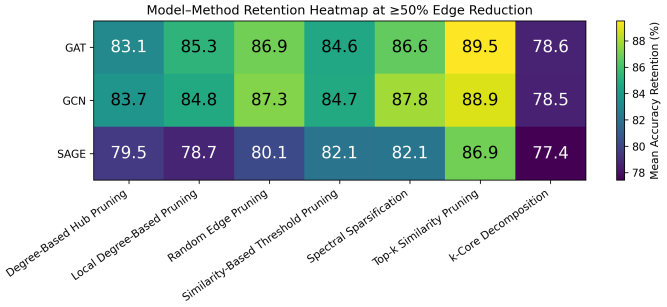

Fig. 9. Mean accuracy retention (%) for each GNN architecture and sparsification method under moderate-to-high sparsity (at least 50% edge reduction). Values are averaged across all benchmark datasets.

task-relevant feature neighborhoods rather than focusing solely on edge-count reduction.

VI. Conclusion and Future Work

In this paper, we presented a comprehensive benchmark for evaluating the robustness of Graph Neural Networks under graph sparsification. Our results demonstrate that preserving task-relevant structural and semantic relationships, rather than simply maintaining graph connectivity, is critical for maintaining predictive performance. Among the evaluated methods, feature-aware sparsification consistently provides the strongest robustness, while spectral sparsification offers an effective structure-preserving alternative. The study also reveals that robustness depends both on the underlying GNN architecture and on the characteristics of the graph data set, highlighting that no single sparsification strategy is universally optimal.

Overall, our findings show that substantial graph compression is achievable with only limited loss in predictive performance when informative structural and semantic relationships are preserved. We hope that this benchmark serves as a useful reference for designing robust and efficient graph learning systems. Future work will extend the evaluation to larger real-world graphs, additional graph learning tasks, graph transformers, and adaptive task-aware sparsification methods.

Acknowledgment

This material is based upon work supported by the National Science Foundation (NSF) under Award Number 2323533.

References

[1] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International conference on machine learning*. PMLR, 2016, pp. 40–48.

[2] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.

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

[4] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He *et al.*, "A survey of graph neural networks for recommender systems: Challenges, methods, and directions," *ACM Transactions on Recommender Systems*, vol. 1, no. 1, pp. 1–51, 2023.

[5] V. Induru and G. Arulkumaran, "Adaptive cybersecurity monitoring via semantic stream processing and gnn-based trust scoring on ipv4 logs," *International Journal of Business Management and Economic Review*, vol. 4, no. 4, p. 430, 2021.

[6] G. Muzio, L. O'Bray, and K. Borgwardt, "Biological network analysis with deep learning," *Briefings in bioinformatics*, vol. 22, no. 2, pp. 1515–1530, 2021.

[7] A. Sharma, P. K. Singh, P. Nikashina, V. Gavrilenko, A. Tselykh, and A. Bozhenyuk, "Ai and gnn model for predictive analytics on patient data and its usefulness in digital healthcare technologies," in *IoT, Big Data and AI for Improving Quality of Everyday Life: Present and Future Challenges: IOT, Data Science and Artificial Intelligence Technologies*. Springer, 2023, pp. 331–345.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[9] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[10] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[11] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[12] R. Hassan and S. Arifuzzaman, "An efficient multi-core parallel implementation of sssp algorithm with decreasing delta-stepping," in *2024 IEEE High Performance Extreme Computing Conference (HPEC)*, 2024, pp. 1–7.

[13] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *Proceedings of the ACM SIGKDD workshop on mining data semantics*, 2012, pp. 1–8.

[14] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," *arXiv preprint arXiv:1907.10903*, 2019.

[15] W. Ju, S. Yi, Y. Wang, Z. Xiao, Z. Mao, H. Li, Y. Gu, Y. Qin, N. Yin, S. Wang *et al.*, "A survey of graph neural networks in real world: Imbalance, noise, privacy and ood challenges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[16] D. A. Spielman and N. Srivastava, "Graph sparsification by effective resistances," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008, pp. 563–568.

[17] D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig, "Sparsification—a technique for speeding up dynamic graph algorithms," *Journal of the ACM (JACM)*, vol. 44, no. 5, pp. 669–696, 1997.

[18] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 257–266.

[19] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Graphsaint: Graph sampling based inductive learning method," *arXiv preprint arXiv:1907.04931*, 2019.

[20] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. Pmlr, 2017, pp. 1263–1272.

[21] U. Alon and E. Yahav, "On the bottleneck of graph neural networks and its practical implications," *arXiv preprint arXiv:2006.05205*, 2020.

[22] M. Hamann, G. Lindner, H. Meyerhenke, C. L. Staudt, and D. Wagner, "Structure-preserving sparsification methods for social networks," *Social Network Analysis and Mining*, vol. 6, no. 1, p. 22, 2016.

[23] J. Han, W. Huang, Y. Rong, T. Xu, F. Sun, and J. Huang, "Structure-aware dropedge toward deep graph convolutional networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 11, pp. 15 565–15 577, 2024.

[24] S. B. Seidman, "Network structure and minimum degree," *Social networks*, vol. 5, no. 3, pp. 269–287, 1983.

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

[26] Y. Chen, L. Wu, and M. Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," *Advances in neural information processing systems*, vol. 33, pp. 19 314–19 326, 2020.