Distribution-Aware Black-Box Graph Injection Attack via Local Neighborhood Context

Chengyu Zhu[†,1], Weina Xu[†,1], Yufei Hou[*,1], Kahim Wong[2], Zeming Liu[1], Jiantao Zhou[2], Yuanfang Guo[1]
[1] *School of Computer Science and Engineering, Beihang University, Beijing, China*
[2] *Department of Computer and Information Science, University of Macau, Macau, China*
Email: 23371543@buaa.edu.cn, weinaplus@gmail.com, houyufly@163.com, yc37437@um.edu.mo, zmliu@buaa.edu.cn, jtzhou@um.edu.mo, andyguo@buaa.edu.cn

***Abstract*—Graph injection attacks inject carefully crafted malicious nodes and edges into graph-structured data, posing a serious threat to Graph Neural Networks (GNNs) whose predictions heavily rely on topological aggregation. As GNNs are widely deployed in real-world applications, such as recommendation systems and security-sensitive domains, investigating such attacks is essential for assessing practical risks and improving the reliability of GNN-based systems. Existing methods have been proposed to analyze the vulnerability of GNNs. However, they still struggle to balance attack effectiveness and imperceptibility. To address this challenge, we propose DAGIA, a novel Distribution-Aware Graph Injection Attack framework that leverages local neighborhood context to achieve imperceptible and effective node injections. Specifically, DAGIA firstly identifies vulnerable nodes via a multi-metric assessment. Guided by Neighborhood Label Distribution (NLD), DAGIA then constructs injection topology by clustering vulnerable nodes and initializes injected node features. Finally, a margin-based optimization refines injected features to enhance attack performance. Extensive experiments on benchmark datasets demonstrate that DAGIA achieves high attack effectiveness while maintaining strong imperceptibility, outperforming existing state-of-the-art methods. The source code of this work is available at https://github.com/yizhidahuangxiong/DAGIA.**

***Index Terms*—graph neural networks, graph injection attack, adversarial attacks**

I. Introduction

Graph Neural Networks (GNNs) [1] have emerged as a powerful paradigm for learning from graph-structured data and have achieved remarkable success across a wide range of applications [2], [3], including social networks, recommendation systems, and security-critical domains such as fraud detection and intrusion analysis. By leveraging neighborhood aggregation, GNNs effectively capture complex relational patterns, but this design also makes them vulnerable to adversarial attacks [4], which can significantly degrade prediction performance and undermine model reliability.

Graph Injection Attacks (GIAs) [5] are a typical form of adversarial attacks on GNNs, where carefully crafted malicious nodes and edges are injected into the graph to manipulate neighborhood aggregation and disrupt model predictions. Unlike traditional graph modification attacks (GMAs) that alter existing nodes or edges, GIAs introduce new components that interact with target nodes and propagate adversarial influence through the graph. This paradigm is particularly effective for GNNs, as injected nodes can exert strong influence while remaining statistically consistent with the original graph. Compared to GMAs, GIAs are often more practical, enabling stealthy attacks under limited budgets or restricted access scenarios.

Existing studies have proposed various graph injection attacks to investigate the vulnerabilities of GNNs. Early works generate malicious nodes by jointly optimizing features and connections via gradient-based methods [5] or parametric models [6]. While effective in inducing misclassification, these attacks often violate graph homophily, which significantly compromises structure imperceptibility. Recent works incorporate heuristics to improve stealthiness and attack effectiveness. TDGIA [7] identifies vulnerable low-degree targets using topological deficit, and LPGIA [8] leverages label propagation dynamics to manipulate global predictions. Although they enhance attack effectiveness under specific assumptions, they often neglect the semantic distribution of node features or rely on architecture-specific propagation, limiting transferability and detectability resistance. TANIA [9] is a topology-aware node injection attack that uses a two-stage topology construction strategy and adaptive feature optimization to achieve targeted misclassification. While effective, the reliance on discrete best-wrong-class (BWC) labels and prototype-based neighbor averaging limits grouping granularity and feature imperceptibility, which reduces attack strength under fixed injection budgets. Despite their diverse designs and notable progress, existing graph injection attack methods still struggle to simultaneously achieve strong attack effectiveness and high imperceptibility.

To address the above limitations, we propose DAGIA, a novel **D**istribution-**A**ware **G**raph **I**njection **A**ttack that leverages local neighborhood context to achieve both effective and imperceptible node injections. Specifically, DAGIA begins with a multi-metric vulnerable node identification strategy that jointly considers neighborhood feature smoothness, local feature variance, and structure homophily, enabling the selection of targets that are both semantically influential and structurally consistent under limited injection budgets. Guided by the Neighborhood Label Distribution (NLD), DAGIA then performs cluster-based topology construction, grouping vulnerable nodes with similar neighborhood label patterns and assigning each

[†] These authors contributed equally to this work.
[*] Corresponding author.

group to a shared injected node. This distribution-aware grouping allows injected nodes to influence multiple targets with aligned aggregation contexts, improving attack effectiveness while preserving structure plausibility. Based on the same NLD information, DAGIA further introduces a distribution-aware feature initialization scheme, where injected node features are initialized using distractor class prototypes derived from group-level neighborhood statistics. This design anchors the injected nodes within the existing semantic distribution of the graph, avoiding conspicuous feature outliers. Finally, a margin-based feature optimization refines the injected features to induce misclassification by minimizing decision margins. Our main contributions are summarized as follows:

- We propose DAGIA, a distribution-aware framework for node injection attacks that achieves strong adversarial performance while preserving imperceptibility. DAGIA consists of four main stages, including vulnerable node identification, topology generation, injected prototype initialization, and feature optimization, forming a general pipeline for attacking GNNs.
- DAGIA employs Neighborhood Label Distribution (NLD) to jointly guide topology construction and injected feature initialization. By clustering vulnerable nodes based on NLD and selecting injected node prototypes from the aggregated group-level distribution, DAGIA ensures that injected nodes align with shared neighborhood semantics, enabling effective attacks while preserving imperceptibility.
- We conduct comprehensive evaluations on 6 real-world datasets and 6 GNN architectures, showing that DAGIA consistently outperforms state-of-the-art attackers in both attack effectiveness and imperceptibility.

II. RELATED WORK

Graph injection attacks constitute an adversarial paradigm for GNNs by introducing malicious nodes and edges. Some works [5] estimate optimization directions by exploiting gradient signals obtained from the target model or an approximated surrogate, allowing injected nodes to be crafted in a more targeted way. Building on this, subsequent research [6] leverages these gradient signals to train a generative model for producing adversarial components. Other studies adopt diverse paradigms, such as reinforcement learning [10], [11] and heuristic-based [7], [8], [12] strategies, to identify effective injection actions under different graph structures and attack constraints. TANIA [9] introduces a two-stage topology construction and an adaptive feature optimization module for efficient structure design and high-quality feature generation of injected nodes. However, existing graph injection attacks still struggle to balance strong attack performance with imperceptibility, highlighting the necessity for more effective injection strategies.

III. PRELIMINARIES

A. Graph Injection Attacks

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a graph with n nodes, characterized by an adjacency matrix $\mathbf{A} \in \{0,1\}^{n\times n}$ and a feature matrix $\mathbf{X} \in \mathbb{R}^{n\times F}$. Graph Injection Attacks compromise GNNs by introducing a small number of carefully crafted nodes and edges, which influence neighborhood aggregation and consequently disrupt model predictions.

Specifically, GIAs introduce a set of injected nodes $\mathcal{V}_{inj}$ and edges $\mathcal{E}_{inj}$ into the original graph $\mathcal{G}$, resulting in a perturbed graph $\mathcal{G}' = (\mathcal{V}\cup\mathcal{V}_{inj}, \mathcal{E}\cup\mathcal{E}_{inj})$. The adversary aims to optimize the injected features $\mathbf{X}_{inj}$ and $\mathcal{E}_{inj}$ to maximize performance degradation on a target node set $\mathcal{V}_{target}$.

B. Threat Model

We consider a non-targeted black-box node injection attack under the evasion setting, where the attacker has no access to the internal parameters of the victim GNN. We aim to maximize the overall misclassification rate under a global attack objective, subject to a node injection budget $\Delta_{|\mathcal{V}_{inj}|} = rn$ and an edge budget Δ_e. Here, r is the injection ratio. $\Delta_{|\mathcal{V}_{inj}|}$ limits the total number of injected nodes, while Δ_e restricts the number of edges connected to each injected node.

IV. PROPOSED METHOD

In this section, we propose DAGIA, a novel **D**istribution-**A**ware **G**raph **I**njection **A**ttack framework that leverages Neighborhood Label Distribution (NLD) to achieve imperceptible and effective node injections. As illustrated in Fig. 1, DAGIA consists of four stages: vulnerable node identification (Section IV-A), topology generation (Section IV-B), injected prototype initialization (Section IV-C), and feature optimization (Section IV-D), forming a general pipeline for attacking GNNs.

A. Multi-Metric Vulnerable Node Identification

DAGIA firstly requires identifying the most vulnerable target nodes to maximize attack effectiveness under a limited injection budget. Unlike traditional ways that rely solely on surrogate models or topology, we also incorporate intrinsic semantic information of nodes, as neglecting such information often reduces transferability and limits generalization across different graphs. To address this, we propose a multi-metric assessment that combines intrinsic graph statistics with model-based predictions. These three types of metrics are described as follows.

(1) **Feature Smoothness** S_f**.** Since GNNs rely on the message-passing mechanism, nodes with high feature similarity to their neighbors are subjected to stronger smoothing effects, making them more sensitive to perturbations in the aggregation scope. Thus, we quantify the smoothness by calculating the cosine similarity between the feature $\mathbf{X}_v$ of the node v and the mean feature $\bar{\mathbf{X}}_{\mathcal{N}(v)}$ of its neighbors, i.e.,

$$S_f(v) = \frac{\mathbf{X}_v \cdot \bar{\mathbf{X}}_{\mathcal{N}(v)}}{\|\mathbf{X}_v\|\|\bar{\mathbf{X}}_{\mathcal{N}(v)}\|}, \tag{1}$$

A higher $S_f(v)$ indicates better alignment with the neighbor context, enabling injected nodes to effectively influence the target's representation.

(2) **Neighborhood Noise** S_n**.** The imperceptibility of injected nodes is influenced by the neighborhood of the selected

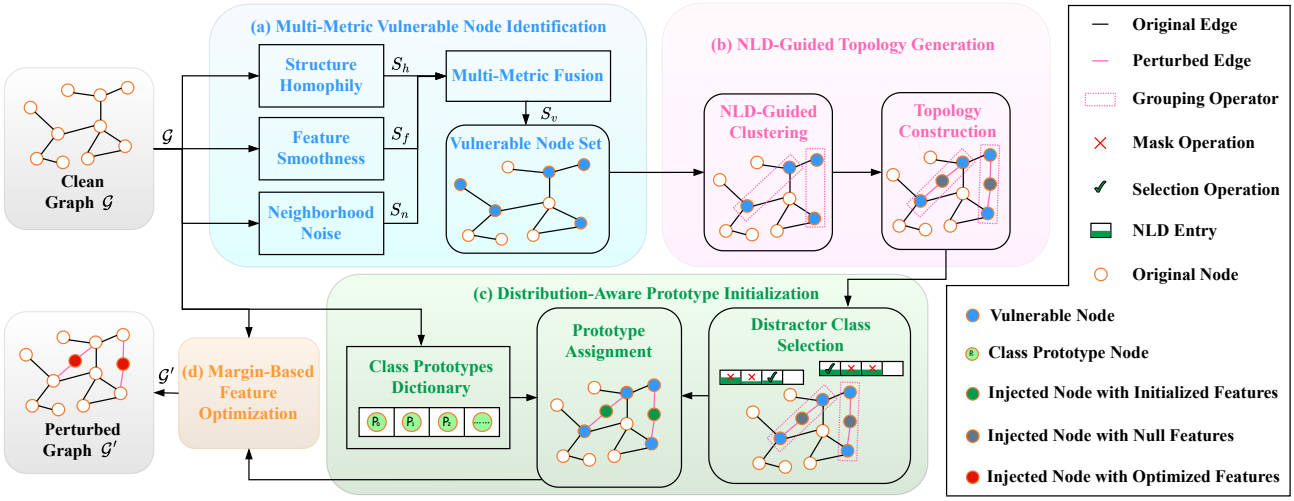

Fig. 1. The overall framework of DAGIA. (a) Multi-metric vulnerable node identification: selects target nodes. (b) NLD-guided topology generation: partitions targets for injection. (c) Distribution-aware prototype initialization: selects class prototypes for injected nodes. (d) Margin-based feature optimization: refines injected features to induce misclassification.

vulnerable target node. Injected nodes are less likely to be detected when the target node's neighborhood exhibits high feature variance, as the feature distribution makes injected perturbations less distinguishable. Furthermore, lower-degree nodes are preferred as their smaller neighborhoods allow injected perturbations to exert a stronger influence. Thus, we define the neighborhood noise of a node v as follows.

$$S_n(v) = \frac{\sum_{k=1}^{F} \mathrm{Var}_k(\mathcal{N}(v))}{\sqrt{d_v + 1}}, \tag{2}$$

where $\mathrm{Var}_k(\mathcal{N}(v))$ denotes the variance of the k-th feature in the neighborhood, and $\sqrt{d_v+1}$ is a normalization factor.

(3) **Structure Homophily** S_h**.** Since GNNs heavily rely on graph homophily, perturbing high-homophily nodes yields the greatest impact on overall accuracy. Thus, we prioritize them as targets. We leverage a surrogate model f_{surr} to obtain predictions $\hat{y}$ and define a node's structure homophily as the proportion of neighbors with the same predicted label, i.e.,

$$S_h(v) = \frac{1}{d_v} \sum_{u \in \mathcal{N}(v)} \mathbb{I}(\hat{y}_u = \hat{y}_v), \tag{3}$$

where $\mathbb{I}(\cdot)$ is the indicator function.

After obtaining the above three metrics, we compute the final vulnerability score for each node as follows.

$$S_v(v) = \mathcal{Z}(S_f(v)) + \mathcal{Z}(S_n(v)) + \beta \cdot \mathcal{Z}(S_h(v)), \tag{4}$$

where $\mathcal{Z}(\cdot)$ denotes Z-score normalization, and β is a hyper-parameter. Finally, the top-K nodes with the highest S_v are selected as vulnerable target nodes $\mathcal{V}_{\text{target}}$ for injection attacks.

B. NLD-Guided Topology Generation

To efficiently allocate the injection budget, we partition target nodes into multiple groups, such that each group can be effectively attacked by a shared injected node. We use the Neighborhood Label Distribution (NLD) [13], [14] as the signature of the neighborhood to characterize aggregation contexts. NLD describes the distribution of class labels within a node's local neighborhood. Nodes with similar NLDs tend to produce similar aggregated features in GNNs, making them harder to distinguish, whereas nodes with distinct NLDs are easier to separate. Therefore, grouping targets by NLD similarity enables the injected node to induce consistent perturbations across the group, thereby improving attack effectiveness under limited budgets.

NLD-Guided Clustering. We firstly compute the NLD vector $P_v \in \mathbb{R}^C$ for each selected vulnerable node v. Here, C is the total number of labels. Specifically, let $\hat{y}_u$ be the label of neighbor u predicted by the surrogate model f_{surr}. We represent each neighbor's label as a one-hot vector $\mathbf{e}_u \in \{0,1\}^C$, where the $\hat{y}_u$-th element is 1 and all other elements are 0. The NLD is then calculated by averaging these one-hot vectors over the neighborhood.

$$P_v = \frac{1}{d_v} \sum_{u \in \mathcal{N}(v)} \mathbf{e}_u \tag{5}$$

We then apply K-Means clustering on P, with the number of clusters set to the injection budget $\Delta_{|\mathcal{V}_{inj}|}$. This yields cluster centroids $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_{\Delta_{|\mathcal{V}_{inj}|}}\}$, which serve as reference centers for partitioning the target nodes. Based on the clustering results, we assign one injected node $\mathcal{V}_{inj}^v$ to each cluster to serve all target nodes within that cluster. This design allows a single injected node to simultaneously influence a group of targets that share similar aggregation contexts, thereby improving attack effectiveness.

Greedy Grouping. However, the direct assignment may violate the maximum edge constraint Δ_e. To address this issue, we adopt a global greedy allocation strategy. Specifically, we compute the Euclidean distances between the NLD vectors of all target nodes and all NLD centroids. The target–centroid pairs are then sorted in ascending order of distance. We iteratively connect each target node to the corresponding injected node only if this node has not yet been assigned

and the injected node's degree remains below Δ_e, ensuring that the connections respect the maximum edge constraint while prioritizing distributional alignment.

C. Distribution-Aware Prototype Initialization

Once the injected nodes $\mathcal{V}_{inj}$ are connected to the original graph, forming the topology $\mathcal{E} \cup \mathcal{E}_{inj}$, DAGIA initializes the features of the injected nodes. Feature initialization is crucial for maintaining imperceptibility. Random initialization generates statistical outliers that are easily detected, while simple neighbor averaging produces over-smoothed features that lack adversarial impact. To overcome these issues, we define a distractor class for each selected node, which DAGIA then emulates via prototype mimicry to initialize the injected node features.

Distractor Class Definition. For the i-th cluster and its assigned i-th injected node, we firstly compute the average NLD of the cluster, denoted as $\mathbf{P}^i_{ave}$. Since NLD is derived from one-hot encoded labels, $\mathbf{P}^i_{ave}$ reflects the empirical class frequencies within the union of the cluster's neighborhoods. To enhance adversarial effectiveness, we mask the dominant predicted label $\hat{c}_{dom}$ and define the distractor class c^* as the most frequent remaining class, formulated as follows.

$$c^* = \arg \max_{c \neq \hat{c}_{dom}} (\mathbf{P}^i_{ave}(c)) \tag{6}$$

This strategy ensures c^* serves as a natural distractor, inducing a directional bias toward incorrect decision boundaries while maintaining structure stealth.

Prototype Feature Initialization. We initialize the feature of the injected node using the global class prototype $\mathbf{T}_{c^*}$, defined as the mean feature embedding of all nodes predicted as class c^* by the surrogate model f_{surr}. To ensure statistical diversity among the injected nodes, we add small Gaussian noise ϵ to $\mathbf{T}_{c^*}$, and the initialized feature of the i-th injected node is formulated as follows.

$$\mathbf{X}^i_{inj} = \mathbf{T}_{c^*} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{I}) \tag{7}$$

D. Margin-Based Feature Optimization

To further refine the initialized features and enhance attack effectiveness, DAGIA adopts the Carlini-Wagner (CW) loss [15] to optimize $\mathbf{X}_{inj}$. Accordingly, the objective of DAGIA is formulated as,

$$\mathcal{L}_{CW} = \frac{\sum\limits_{v \in \mathcal{V}_{\text{target}}} \max\left(Z_{\hat{y}_v}(v) - \max\limits_{j \neq \hat{y}_v} Z_j(v) + \kappa, 0 \right)}{|\mathcal{V}_{\text{target}}|} \tag{8}$$

where $Z_*(v)$ denotes the logit output of the surrogate model f_{surr} for node v regarding class $*$, $\hat{y}_v$ denotes the predicted label, and κ is the confidence parameter.

V. EXPERIMENTS

A. Experimental Setup

Datasets. We evaluate the attack performance of the proposed DAGIA on 6 benchmark datasets: GRB-Cora, GRB-Citeseer [16], PubMed [17], BlogCatalog [18], Ogbn-Arxiv [19], and Actor [20].

Victim Models. Two popular types of GNNs are selected as victim models: (1) General GNNs, including GCN [21], GraphSAGE [22], GAT [23], and H2GCN [24]; (2) Robust GNNs, including EGNNGuard [25] and RobustGCN [26].

Baselines. We systematically compare DAGIA with existing state-of-the-art black-box graph injection attack methods, including SPEIT [16], TDGIA [7], RTDGIA [12], LPGIA [8], and TANIA [9].

Implementation Details. In our experiments, we adopt a standard GCN as the surrogate model of DAGIA, and set $\beta = 2.0$. For fair comparison, we set the node injection budget $\Delta_{|\mathcal{V}_{inj}|} = 0.01n$ and fix the edge budget Δ_e to the average degree of each dataset. For all datasets, we follow the official train/validation/test splits provided by their respective sources. All experiments are conducted on an NVIDIA RTX 4060 GPU. We report the average results over 5 independent runs.

B. Attack Performance Comparison

We analyze the attack performance of DAGIA and compare it with state-of-the-art black-box attacks. Table I presents the experimental results.

Attack Performance. Our DAGIA achieves consistently better performance than competing methods across diverse GNNs on most datasets. For homophilic models such as GCN, GraphSAGE and GAT, DAGIA causes the largest performance degradation, achieving an average accuracy reduction of 5.56% relative to the strongest baseline over all datasets. This advantage further extends to the heterophily-specific H2GCN, where DAGIA yields the best performance across all datasets. For robust GNNs, DAGIA achieves the greatest performance degradation on RobustGCN across most datasets and performs competitively against EGNNGuard, producing the lowest accuracy on PubMed. These results indicate that the proposed DAGIA effectively exploits fundamental GNN vulnerabilities that are not limited to specific architectures.

Attack Generalization. The results indicate that DAGIA achieves powerful attack generalization across graphs of different scales and types. For citation networks GRB-Cora, DAGIA achieves the lowest accuracy, outperforming TANIA by an average of 5.60%. For heterophilic Actor dataset, it achieves best results across most victim models. Moreover, DAGIA demonstrates strong scalability and effectiveness, outperforming TANIA by 8.60% on Ogbn-Arxiv, and reducing the average accuracy of all victim models by 27.62% on the dense BlogCatalog network.

C. Ablation Studies

We verify the effect of key components of DAGIA via ablation experiments. We present the following variants to observe the attack performance: (1) **'w/o Identification'**, which does not use Multi-Metric Vulnerable Node Identification Module and instead randomly selects nodes as target nodes. (2) **'w/o Topology'**, which does not use a clustering algorithm but instead uses random grouping; (3) **'w/o Initialization'**,

TABLE I

CLASSIFICATION ACCURACY (%) OF VICTIM GNNS UNDER GRAPH INJECTION ATTACKS. 'CLEAN' DENOTES ACCURACY ON THE UNPERTURBED GRAPH. '#AVG.' DENOTES THE AVERAGE ATTACK PERFORMANCE CROSS ALL VICTIM MODELS. **BOLD** AND UNDERLINE INDICATE THE BEST AND SECOND-BEST ATTACKS, RESPECTIVELY. ONLY THREE BASELINES ARE REPORTED ON OGBN-ARXIV DUE TO OOM ISSUES.

Dataset	Victim Model	Clean	SPEIT	TDGIA	RTDGIA	LPGIA	TANIA	DAGIA (Ours)
GRB-Cora	GCN	84.38	76.34	73.58	77.24	83.48	70.95	**57.91**
	GraphSAGE	82.21	78.23	74.75	77.86	81.59	74.23	**68.86**
	GAT	84.60	80.22	76.00	78.26	83.46	74.58	**66.67**
	H2GCN	75.97	71.74	69.38	73.13	75.27	67.51	**62.31**
	RobustGCN	84.40	80.50	78.38	82.51	82.99	76.64	**72.06**
	EGNNGuard	85.15	85.15	84.65	**78.96**	84.18	82.41	84.90
	#Avg.	82.79	78.70	76.12	77.99	81.83	74.39	**68.79**
GRB-Citeseer	GCN	74.59	70.66	67.61	69.03	74.59	66.56	**58.64**
	GraphSAGE	68.90	66.77	63.66	65.06	68.55	62.84	**59.87**
	GAT	73.06	70.78	67.96	69.40	72.62	66.71	**61.78**
	H2GCN	74.23	71.20	69.40	72.60	73.88	67.06	**64.83**
	RobustGCN	75.46	73.90	70.05	73.83	75.09	68.94	**67.71**
	EGNNGuard	73.94	73.94	73.94	69.51	73.86	**68.48**	68.67
	#Avg.	73.36	71.21	68.77	69.91	73.10	66.77	**63.58**
PubMed	GCN	85.80	79.40	80.79	82.86	84.28	78.52	**71.57**
	GraphSAGE	82.66	81.19	79.43	81.44	82.19	**77.86**	78.11
	GAT	84.42	81.86	82.07	82.83	82.99	81.11	**76.24**
	H2GCN	86.03	76.11	81.31	85.47	85.88	76.86	**70.31**
	RobustGCN	86.37	85.05	83.06	83.98	84.46	**81.62**	82.06
	EGNNGuard	85.53	80.57	82.10	82.70	85.37	82.00	**78.41**
	#Avg.	85.14	80.70	81.46	83.21	84.20	79.66	**76.12**
BlogCatalog	GCN	79.02	15.00	21.66	21.73	78.65	15.06	**4.81**
	GraphSAGE	92.17	83.43	87.66	87.93	92.05	83.38	**76.94**
	GAT	65.78	64.70	46.13	51.15	64.35	58.85	**41.87**
	H2GCN	93.00	92.87	92.97	92.96	92.99	92.73	**92.64**
	RobustGCN	70.61	36.25	55.96	55.82	69.95	36.42	**31.61**
	EGNNGuard	82.87	**49.63**	79.31	50.94	82.98	60.01	69.90
	#Avg.	80.58	56.98	63.95	60.09	80.16	57.74	**52.96**
Actor	GCN	28.00	26.96	26.79	25.47	27.41	23.63	**23.34**
	GraphSAGE	32.83	31.70	31.22	33.20	32.47	28.49	**28.36**
	GAT	27.32	26.63	26.47	26.34	26.78	24.39	**24.07**
	H2GCN	33.22	31.97	32.41	33.58	32.97	31.04	**29.95**
	RobustGCN	29.30	28.80	27.50	27.93	28.36	25.92	**25.62**
	EGNNGuard	30.20	30.29	30.30	29.18	30.24	**28.96**	29.75
	#Avg.	30.15	29.39	29.12	29.28	29.71	27.07	**26.85**
Ogbn-Arxiv	GCN	71.63	67.10	48.51	51.02	71.38	44.74	**30.95**
	GraphSAGE	71.48	67.39	**45.66**	49.79	71.29	47.07	48.01
	GAT	71.51	58.10	39.32	39.87	71.34	37.37	**24.41**
	#Avg.	71.54	64.20	44.50	46.89	71.34	43.06	**34.46**

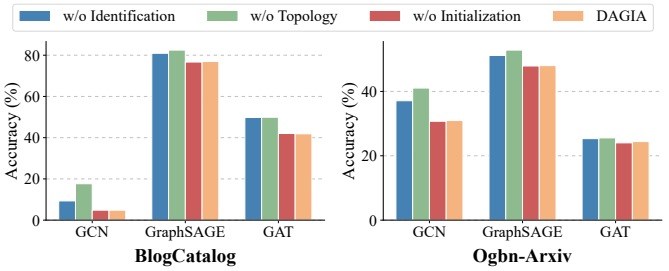

Fig. 2. Ablation Study. Performance comparison of DAGIA and its variants across various GNNs.

which replaces the Prototype Initialization Module with random Gaussian noise for feature initialization.

Fig. 2 shows that vulnerable node identification and NLD-guided topology generation are critical to attack effectiveness, while distribution-aware prototype initialization mainly ensures statistical imperceptibility, with the full DAGIA framework achieving the strongest overall performance.

D. Attack Budget Analysis

We evaluate the adversarial effectiveness of DAGIA on GRB-Cora and GRB-Citeseer, with injection ratios r ranging from 0.0 to 1.0. As illustrated in Fig. 3, DAGIA consistently induces the most significant accuracy degradation across all victim GNNs. These results highlight the effectiveness of our distribution-

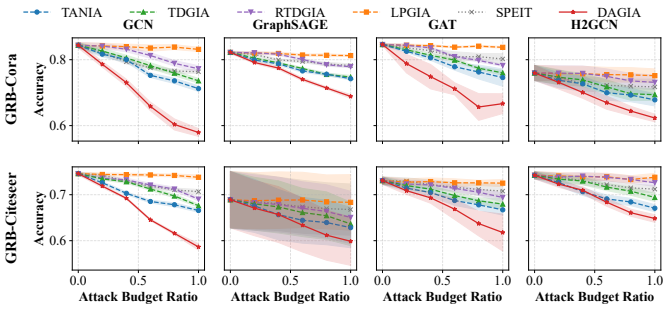

Fig. 3. Comparison of attack performance under different injection rates r. Shaded areas represent standard deviation over five runs.

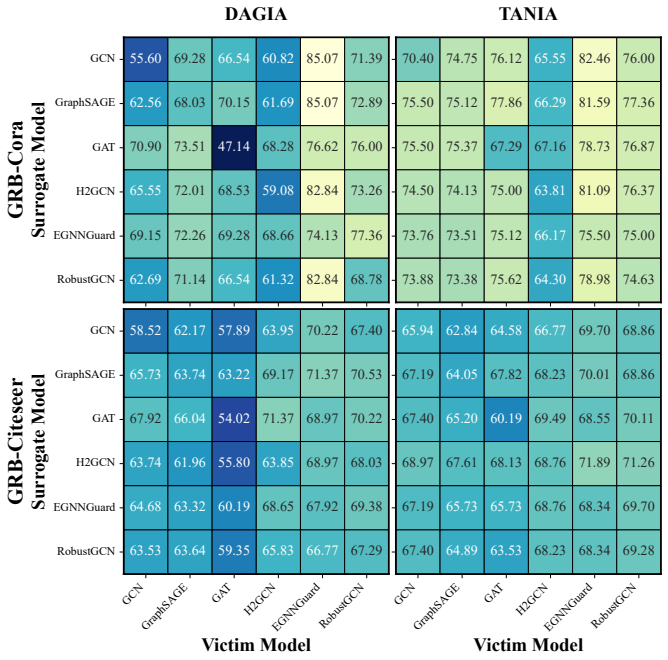

Fig. 4. Transferability of DAGIA vs. TANIA. Cell values represent victim model accuracy.

aware paradigm, showing that DAGIA accurately identifies the most vulnerable structures and maximizes the adversarial impact of injected nodes under a fixed attack budget.

E. Transferability Analysis

To evaluate the transferability of DAGIA, we conduct comparative experiments across diverse surrogate–victim model pairs. As shown in Fig. 4, DAGIA causes greater degradation in classification accuracy than TANIA, indicated by the darker heatmap regions. This demonstrates that leveraging NLD to align local distributions enables DAGIA to capture fundamental structure vulnerabilities that transfer effectively across models.

F. Stealthiness and Distribution Analysis

To evaluate the attack imperceptibility of DAGIA, we measure the discrepancy between the homophily distributions of clean and injected graphs using Kullback-Leibler (KL) divergence [27]. As shown in Fig. 5, the KL divergence of injected graphs generated by DAGIA is significantly lower

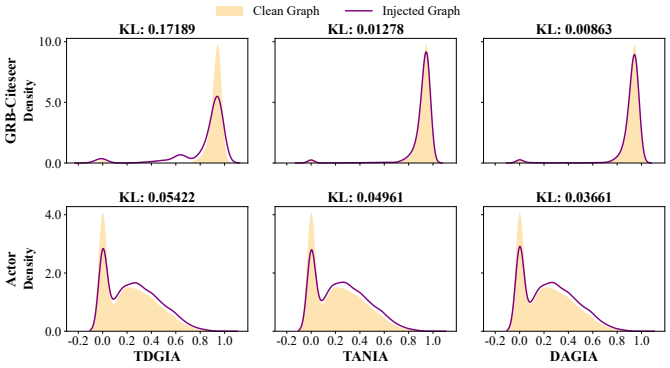

Fig. 5. Homophily distribution comparison of clean vs. injected graphs on GRB-Citeseer and Actor. Lower KL divergence indicates higher statistical imperceptibility.

than that of other baselines, indicating DAGIA maintains high imperceptibility while achieving strong attack effectiveness.

VI. CONCLUSION

In this paper, we propose DAGIA, a novel Distribution-Aware Graph Injection Attack framework that leverages local neighborhood context to achieve imperceptible and effective node injections. DAGIA identifies vulnerable targets via multi-metric analysis, constructs NLD-guided injection topology with distribution-aware feature initialization, and refines injected nodes through margin-based optimization to achieve effective and imperceptible attacks. Extensive evaluations across diverse benchmarks show that DAGIA consistently surpasses state-of-the-art baselines in attack effectiveness and cross-model transferability, while preserving superior statistical imperceptibility.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 62272020, in part by the Hebei Natural Science Foundation (No. F2024202047), in part by Macau Science and Technology Development Fund under 001/2024/SKL, 0119/2024/RIB2, and 0022/2022/A1, in part by State Key Laboratory of Complex & Critical Software Environment (SKLCCSE-2025ZX-23), and in part by the Fundamental Research Funds for Central Universities.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[2] J. Yan, G. Yan, and D. Jin, "Classifying malware represented as control flow graphs using deep graph convolutional neural network," in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2019, pp. 52–63.

[3] Ö. A. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020.

[4] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2018, pp. 2847–2856.

[5] J. Wang, M. Luo, F. Suya, J. Li, Z. Yang, and Q. Zheng, "Scalable attack on graph data by injecting vicious nodes," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1363–1389, 2020.

[6] S. Tao, Q. Cao, H. Shen, J. Huang, Y. Wu, and X. Cheng, "Single node injection attack against graph neural networks," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*, 2021, pp. 1794–1803.

[7] X. Zou, Q. Zheng, Y. Dong, X. Guan, E. Kharlamov, J. Lu, and J. Tang, "TDGIA: Effective injection attacks on graph neural networks," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD)*, 2021, pp. 2461–2471.

[8] P. Zhu, Z. Pan, K. Tang, X. Cui, J. Wang, and Q. Xuan, "Node injection attack based on label propagation against graph neural network," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 5, pp. 5858–5870, 2024.

[9] L. Su, J. Wang, Z. Gan, and D. Li, "Topology-aware node injection attacks against graph neural networks," *Neurocomputing*, vol. 618, p. 129128, 2025.

[10] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, "Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach," in *Proceedings of The Web Conference 2020*, 2020, pp. 673–683.

[11] D. Chen, J. Zhang, Y. Lv, J. Wang, H. Ni, S. Yu, Z. Wang, and Q. Xuan, "Single-node injection label specificity attack on graph neural networks via reinforcement learning," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 5, pp. 6135–6150, 2024.

[12] Y. Chen, H. Yang, Y. Zhang, K. Ma, T. Liu, B. Han, and J. Cheng, "Understanding and improving graph injection attack by promoting unnoticeability," *arXiv preprint arXiv:2202.08057*, 2022.

[13] Y. Ma, X. Liu, N. Shah, and J. Tang, "Is homophily a necessity for graph neural networks?" in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[14] J. Wang, Y. Guo, L. Yang, and Y. Wang, "Understanding heterophily for graph neural networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2024, pp. 50 489–50 529.

[15] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.

[16] Q. Zheng, X. Zou, Y. Dong, Y. Cen, D. Yin, J. Xu, Y. Yang, and J. Tang, "Graph robustness benchmark: Benchmarking the adversarial robustness of graph machine learning," *arXiv preprint arXiv:2111.04314*, 2021.

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

[18] L. Tang and H. Liu, "Relational learning via latent social dimensions," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2009, pp. 817–826.

[19] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 22 118–22 133.

[20] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-GCN: Geometric graph convolutional networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

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

[22] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[23] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[24] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 7793–7804.

[25] X. Zhang and M. Zitnik, "GNNGuard: Defending graph neural networks against adversarial attacks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 9263–9275.

[26] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, "Robust graph convolutional networks against adversarial attacks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2019, pp. 1399–1407.

[27] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.