Structure-Aware Attack on Graph Neural Networks via Imperceptible Node Injection

Journal:	*Transactions on Big Data*
Manuscript ID	TBD-2025-04-0171.R1
Manuscript Type:	Regular Paper
Opposed Editors:	
Keywords:	E.1.d Graphs and networks < E.1 Data Structures < E Data, G.4.g Reliability and robustness < G.4 Mathematical Software < G Mathematics of Computing, I.2.6.g Machine learning < I.2.6 Learning < I.2 Artificial Intelligence < I Computing Methodologies

Structure-Aware Attack on Graph Neural Networks via Imperceptible Node Injection

Fei Yan, Yanlong Tang, Witold Pedrycz, Kaoru Hirota

***Abstract*—Graph neural networks (GNNs) excel in various graph-based tasks due to their exceptional ability to process non-Euclidean data. However, recent research indicates that their predictive performance is highly vulnerable to perturbations from intentionally manipulated data. Current node injection attack methods disrupt GNN training by injecting numerous nodes, often in excessive amounts, making them easily detectable. To address this issue, this study introduces a structure-aware node injection attack (SNIA), which enables effective and subtle attacks under extreme budget constraints. The scheme leverages the network topology to construct an attack candidate set and applies homogeneity constraints to regulate the generation of perturbed features. By eliminating the dependency on surrogate models for generating perturbed data, SNIA effectively diminishes the global classification performance of GNNs based on the network's inherent structure. We conducted attack experiments with the SNIA scheme on real-world network datasets against both general and defensive GNNs. The experimental findings reveal that it significantly surpasses the existing state-of-the-art methods in attack efficiency, while also showcasing outstanding generalization and resilience.**

***Index Terms*—Graph neural networks (GNNs), imperceptible attack, node injection, homophily constraint.**

I. Introduction

GRAPHS serve as powerful abstract mathematical models, effectively capturing complex relationships among entities in the real world [1]. Their non-Euclidean structure makes processing data inherently challenging. Graph neural networks (GNNs) address this challenge by enabling the processing of non-Euclidean data [2], uncovering hidden insights regarding graph data, and applying them effectively to tasks such as node classification [3], link prediction [4], and graph classification [5]. Owing to their computational efficiency and precision, GNNs are increasingly used in fields such as recommendation systems [6], bioinformatics [7], and medical diagnosis support [8]. However, recent studies highlight that GNNs are susceptible to adversarial attacks [9], which can compromise their robustness [10] and exhibit serious implications in areas such as genomic analysis and auxiliary diagnosis in medicine.

Initial perturbations primarily focus on altering the original graph structure to mislead GNNs [11]. These attacks generally manipulate node features and graph topology using gradient-based methods [12], approximate optimization [13], and adversarial exploration attacks [14] to modify graph data. Additionally, some perturbation strategies employ reinforcement learning to invert the node connectivity relationship [15]. Beyond adversarial perturbations that alter graph structures and node features, some approaches focus on modifying true labels through gradient optimization to achieve desired perturbations [16]. However, these methods often lack practical applicability. As an example, in citation networks, attackers face challenges in altering existing citation relationships between articles. Furthermore, modifying the original data requires access to the underlying databases, necessitating significant operational authority from the attacker [17], which limits the feasibility of these techniques.

Node injection attack methods overcome the limitations of traditional graph perturbation methods in real-world applications. These attacks create perturbation data based on the original graph structure, eliminating the need to alter the original graph structure. For instance, generating new profile pages on social networks can form new node relationships without modifying the original data, thereby minimizing the need for elevated operational permissions [18]. Attackers can generate perturbations by using the gradient of the target node loss function [19] and optimization methods [20], similar to altering graph structures. In addition, some studies have proposed poisoning attacks using genetic algorithms [21], Q-learning [17], and topological vulnerabilities [22]. These methods typically involve injecting a certain number of nodes to disrupt GNN training. However, excessive node injection can waste attack resources, and large-scale injections increase the risk of detection by GNN models. Current research indicates that excessive node injection attacks can disrupt graph homophily, enabling robust GNNs to detect structural anomalies and apply appropriate defenses [23]. Extreme cases of such attacks are initially explored in computer vision, where a single pixel injection into an image can misclassify the model [24]. Some studies have explored extreme injection strategies to manipulate the classification of target nodes during GNN training [20], [25]. However, research on reducing global node classification performance in such scenarios is scarce. Additionally, most node injection attack methods rely on building a surrogate model and generating perturbations

This work was supported by the Natural Sciences and Engineering Research Council of Canada (Grant Number: RGPIN-2022-03045).

Fei Yan is with School of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130022, China and Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: yanfei@cust.edu.cn). Corresponding author: Fei Yan.

Yanlong Tang is with School of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130022, China (e-mail: supertyl@outlook.com).

Witold Pedrycz is with Department of Electrical and Computer Engineering, University of Alberta, Edmonton T6G 1H9, Canada (e-mail: wpedrycz@ualberta.ca).

Kaoru Hirota is with School of Computing, Tokyo Institute of Technology, Yokohama 226-8502, Japan and School of Automation, Beijing Institute of Technology, Beijing 100081, China (e-mail: hirota@bit.edu.cn).

using gradient information and optimization processes. These methods necessitate training a separate GNN model, which is time-consuming for practical attacks.

This study examines node injection attack techniques that significantly influence the global prediction outcomes of GNNs under minimal attack resources. We introduce a novel data perturbation approach, named structure-aware node injection attack (SNIA), which leverages the inherent characteristics and structural coherence of graph data to substantially degrade the global classification performance of the target model. Extensive experimental results validate that the proposed SNIA demonstrates high generalizability and resilience, successfully bypassing detection by defensive models. The key contributions of this study are summarized as follows:

1) We propose an imperceptible attack scheme that disrupts GNN training under extreme budget constraints by injecting individual nodes into the original network data, ultimately reducing their overall classification performance.
2) To optimize the effectiveness of the attack, we depart from the conventional approach of using surrogate models for data perturbation. Instead, we employ a macro-level strategy that emphasizes overlapping community assignments for network nodes to create a candidate attack set.
3) To enhance the stealth of this method, we restrict the generated perturbations to maintain strong homophily with the two-hop neighbors of target nodes, thereby minimizing the impact on the original network structure.

The remainder of this paper is organized as follows. Section II reviews GNNs and related research on graph injection attacks. Section III presents a comprehensive discussion of the proposed attack scheme, covering node community affiliation, overlapping community detection, and subtle feature generation. Section IV outlines a series of experiments designed to validate the effectiveness of the proposed method and provides the necessary parameter settings for the reproducibility of the experiments. Finally, Section V concludes the key findings of this study and discusses future directions for the research.

II. Preliminaries

This section defines the key concepts related to GNNs and node injection attacks to clarify the SNIA strategy proposed in this study. Table I lists the standard symbolic notations.

A. Graph Neural Networks (GNNs)

Given an attribute graph $G = (V, E, X, Y)$, where $V \in \{v_1, v_2, \ldots, v_n\}$ implies the set of graph nodes; $E = \{e_{ij} \mid 0 < i, j \leq n\}$ refers to the set of edges; e_{ij} signifies an edge between nodes i and j; $X \in \mathbb{R}^{n \times d}$ denotes the d-dimensional feature matrix of the graph; Y indicates the label set of graph nodes; and $n = |V|$ represents the total number of nodes in the graph. The linking relationships between nodes are described by the adjacency matrix A, where $A_{i,j} = 1$ if an edge exists between nodes i and j, and $A_{i,j} = 0$ otherwise.

In GNN research, the Graph Convolutional Network (GCN) [26] is the most prominent model, commonly featuring a two-layer graph convolutional architecture. The process of updating node representations in the first layer is performed as follows:

TABLE I
Common Notations and Their Descriptions in SNIA

Notations	Descriptions	Notations	Descriptions
G	Original attribute graph	G'	Poisoned attribute graph
V	Node set of G	V'	Node set of G'
E	Edge set of G	E'	Edge set of G'
A	Adjacency matrix of G	A'	Adjacency matrix of G'
X	Feature matrix of G	X'	Feature matrix of G'
$\mathcal{F}_\theta(\cdot)$	GNN classifier	$\triangle$	Total attack budget
$\triangle F$	Feature attack budget	$\triangle E$	Edge attack budget
y	Real node label	$\tilde{y}$	Predicted node label
v	Node of G	p	Injected fake node

$$H^{(l+1)} = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right), \tag{1}$$

where σ signifies the nonlinear activation function ReLU; $H^{(l)}$ refers to the node representation at the l-th layer; $\hat{A} = \tilde{D}^{-\frac{1}{2}}(A + I)\tilde{D}^{-\frac{1}{2}}$ indicates the normalized adjacency matrix; I is the identity matrix; $\tilde{D}$ denotes the degree matrix; and $W^{(l)}$ represents the learnable parameter weight matrix at the l-th layer. After two GCN update layers, the probability distribution of node category is derived as follows:

$$\mathcal{F}_\theta(A, X) = \mathrm{softmax}\left(\hat{A}\sigma(\hat{A}XW^{(0)})W^{(1)}\right), \tag{2}$$

where $\mathcal{F}_\theta(\cdot)$ implies the node classifier. This study seeks to optimize the classification accuracy on the perturbation graph G' using meticulously crafted perturbation data.

B. Node Injection Attacks

In the node injection attack technique, attackers typically inject a set of artificial fake nodes $P = \{p_1, p_2, \ldots, p_m\}$ into the original network, where m denotes the total number of injected nodes. After executing node injection, the perturbed network $G' = (V', E', X')$ is formed. The adjacency matrix of the perturbed graph is constructed by merging the original node connections with those of the perturbed nodes, while the feature matrix is derived by concatenating the features of the perturbed nodes with the original network features. Consequently, the adjacency matrix A' and feature matrix X' of the perturbed graph are specified as follows:

$$A' = \begin{bmatrix} A & A_p \\ A_p^T & B \end{bmatrix}, X' = \begin{bmatrix} X \\ X_p \end{bmatrix}, \tag{3}$$

where A_p signifies the connections between the injection and original network nodes, B indicates the connection relationships among the injection nodes, and $X_p \in \mathbb{R}^{m \times d}$ denotes the feature matrix of the injected nodes.

C. Problem Definition

This study examines node classification tasks, focusing on the impact of minor perturbations on GNN predictive accuracy. Attackers typically inject a series of perturbations into the

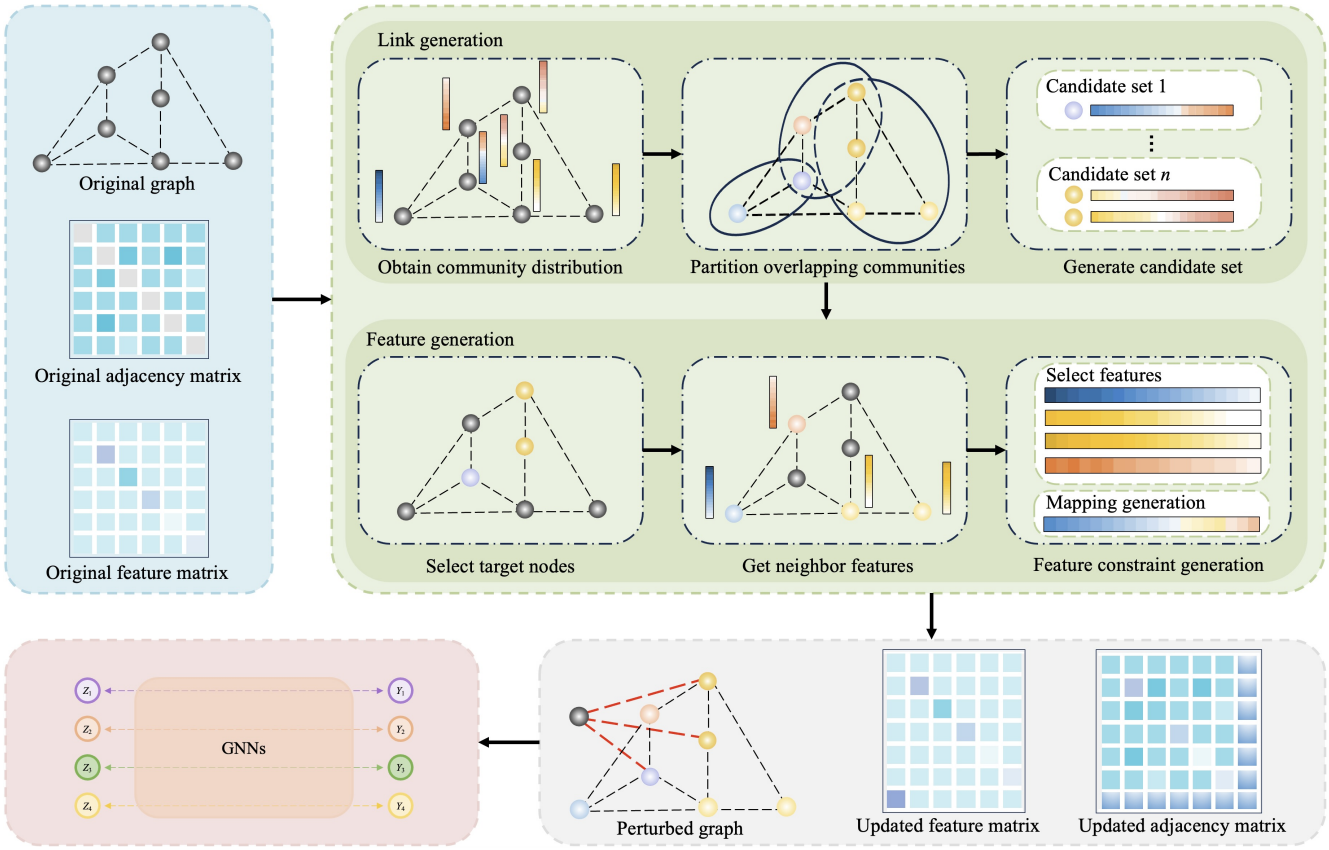

Fig. 1. Framework of the SNIA model. The process consists of three main stages: (1) constructing a set of candidate nodes using the LDA method; (2) selecting target nodes based on a given attack strategy and constructing connections for the poisoned node; and (3) generating features for the poisoned node and injecting a fake node into the original graph to achieve structure-aware perturbation.

original graph to disrupt the training of the target GNN. The task aims to develop a method for generating perturbations, maximizing the degradation of the model's training accuracy. Accordingly, this problem is formally defined as follows:

$$\begin{aligned} &\max \sum_{v=1}^{i} \mathbb{I}\left(\mathcal{F}_{\theta^*}\left(A', X'\right) \neq y_v\right), i \in V_{test} \\ &\text{s.t. } \theta^* = \arg\min_{\theta} \mathcal{L}_{\text{train}}\left(\mathcal{F}_{\theta}\left(A', X'\right)\right), G' - G \leq \Delta \end{aligned}, \quad (4)$$

where V_{test} refers to the test node set; i is the number of nodes in the test set; y_v denotes the true label of node v; $\mathbb{I}$ implies an indicator function that returns 1 if the argument is true and 0 if false; Δ indicates the total attack budget from both feature and link budgets assigned to the injected nodes, measuring the overall changes from the original graph G to the perturbed graph G'; and $\mathcal{L}_{\text{train}}$ represents the training loss, commonly formulated as the cross-entropy function:

$$\mathcal{L}_{\text{train}}(\mathcal{F}_{\theta}(A', X')) = \sum_{v=1}^{j} -y_v \log \tilde{y}_v, j \in V_{train}, \quad (5)$$

where V_{train} signifies the training set of nodes, and j is the number of nodes in the training set. To utilize minor data perturbations and affect the overall prediction accuracy of GNNs, as expressed in Eq. (4), we conducted a comprehensive analysis of the network's structure. The specific methods and implementation details are discussed in Section III.

III. Methodology

This section presents a comprehensive introduction to our proposed node injection attack model, which constructs efficient perturbations by leveraging the intrinsic structural properties of the graph. The scheme encompasses three pivotal components: affinity-based community assignment of nodes (cf. Section III-B), a connection generation strategy for injected nodes (cf. Section III-C), and a stealthy feature construction method (cf. Section III-D).

A. Overall Framework of the SNIA

To effectively illustrate the workflow of SNIA, we present its overall architecture in Fig. 1. A group of densely connected nodes is considered a community, typically composed of entities with similar attributes or types, as noted in [27]. Distinct communities generally exhibit apparent structural separation. The primary objective of GNNs is to classify entities within the same community into a single category. However, in real-world networks, an individual entity may simultaneously belong to multiple communities, a structural property known as overlapping communities [28]. Nodes with such multicommunity tendencies can significantly influence the information propagation process in GNNs due to their involvement in multiple community contexts.

Drawing from this observation, we propose a node injection attack method capable of adequately undermining the

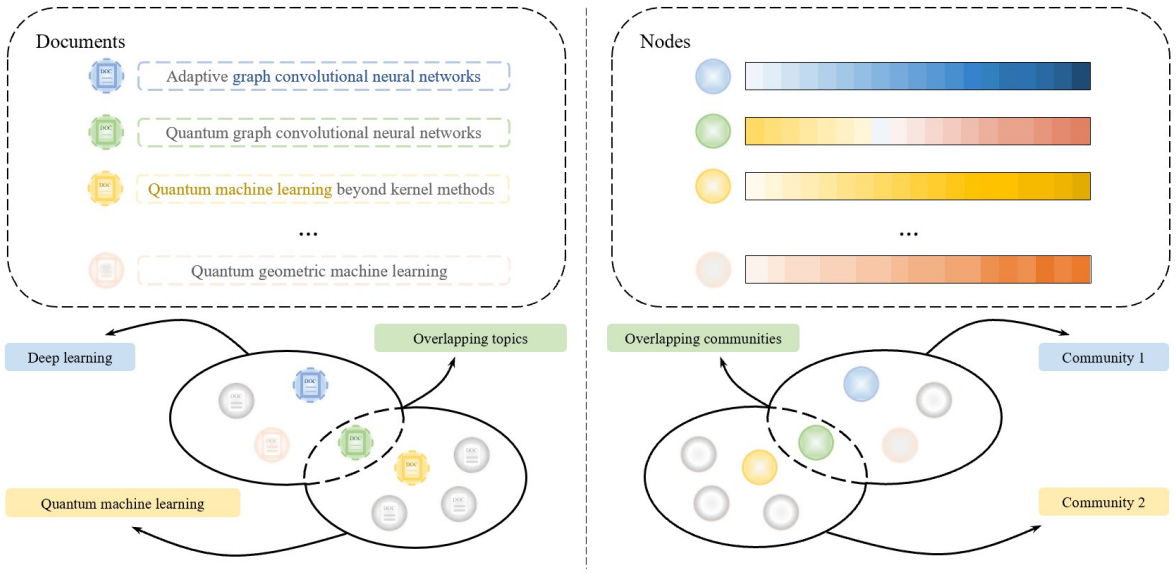

Fig. 2. Comparison of network nodes and their features with the thematic representation of documents and words.

decision accuracy of GNNs under extreme conditions. In the overlapping community detection phase, inspired by the Latent Dirichlet Allocation (LDA) method, commonly used in natural language processing to model "document–word" relationships [29], we analogize graph nodes to "documents" and their attribute features to "words." This analogy enables us to learn each node's distribution over a latent "semantic topic" space, from which we infer its community affinities and construct a candidate set of nodes for potential attacks. Additionally, based on the structural properties of the graph, we design three strategies to generate connections between fake nodes and real nodes. These strategies fully leverage the feature differences among the candidate nodes to enhance the overall perturbation effect. Finally, grounded in the message-passing mechanism of GNNs and incorporating a structural homophily constraint, we develop an effective feature generation approach for fake nodes that is applicable across various scenarios.

In summary, SNIA serves as a versatile data perturbation scheme that generates adversarial data prior to GNN training. As such, it is independent of specific GNN architectures, making it highly transferable across diverse models. The framework consists of two main stages: the data perturbation and the adversarial evaluation. During the data perturbation stage, we create fake nodes with disruptive features designed to interfere with the GNN training process. In the adversarial evaluation stage, we evaluate the effects of the perturbed data on the training of arbitrary GNN models. The implementation details for each module will be discussed in the following subsections.

B. Community Affinity Distribution

Earlier studies utilized gradient information from surrogate models or meta-learning frameworks to determine fake node connections. However, these techniques generate only a single perturbed edge per iteration, which results in considerable time and space complexities [10], [18], rendering them impractical for real-world applications. To overcome these limitations, we abandon surrogate model-based perturbation generation and instead derive perturbation information directly using the structural properties of the network.

To maximize the impact of perturbed data on the decision accuracy of GNNs, we expand our data perturbation strategy to involve connections with multiple nodes that exhibit multi-community preferences. As illustrated in Fig. 2, the structural relationships in real-world networks are often intricate—an entity with diverse preferences can simultaneously belong to multiple communities. Drawing inspiration from the "document–word" analogy, we integrate the LDA model into the graph structure to uncover the overlapping community structures among nodes. Within this modeling framework, the likelihood function for the generative process of a specific node i is defined as follows:

$$\mathcal{L} = \prod_{i=1}^{n} \int p\left(\theta_i|\alpha\right) \prod_{j=1}^{d} \sum_{k_{ij}} p\left(k_{ij}|\theta_i\right) p\left(x_{ij}|k_{ij},\beta\right) d\theta_i, \quad (6)$$

where $p\left(\theta_i|\alpha\right)$ signifies the community distribution for node i, following a Dirichlet distribution with α representing its hyperparameter; $p\left(k_{ij}|\theta_i\right)$ indicates the community distribution for the j_{th} feature of node i; k_{ij} represents the community corresponding to the j_{th} feature of node i; $p\left(x_{ij}|k_{ij};\beta\right)$ refers to the probability of feature x_{ij} being generated by community k_{ij}, where x_{ij} is the j_{th} feature of node i, and β denotes the parameter for the feature distribution of each community.

To approximate the intractable posterior $p(\theta, k \mid X)$, we employ variational inference with the following variational distribution:

$$q(\theta_i, k_i) = q(\theta_i|\gamma_i) \prod_{j=1}^{d} q(k_{ij}|\phi_{ij}), \quad (7)$$

Algorithm 1: Execution of the SNIA strategy

Input: The original attribute graph $G = (V, E, X, Y)$, community distribution parameter k, attack budget $\triangle$, candidate set T, overlapping community set C, and community affinity parameter ζ
Output: Poisoned graph $G' = (V', E', X')$

Initialize the undirected network G
Select the largest connected component in the original network
$\triangle E \leftarrow$ Compute the average degree of the network nodes
$\triangle F \leftarrow$ Determine the average value of non-zero features
/* Calculate the community distribution for all nodes n in the original graph. */
for *each* $i \in [0, n-1]$ **do**
  $\mathcal{L} \leftarrow$ Calculate the likelihood function using Eq. (7)
  $\mathcal{L} \leftarrow$ Optimize the calculation through variational distribution
  $\theta_{ik} \leftarrow$ Calculate the k-community distribution of nodes using Eq. (9)
end for
/* Initialize the overlapping communities and candidate node sets. */
$C \leftarrow 0$
$T \leftarrow 0$
if $\theta_{ik} > \zeta$ **then**
  $C \leftarrow$ Identify overlapping communities using Eq. (10)
  $T \leftarrow$ Assign nodes from the overlapping communities to the candidate set
end if
$V' \leftarrow$ Insert fake nodes into the original node set
$E' \leftarrow$ Generate perturbed edges using the attack method
/* Initialize the homophily parameters. */
$H \leftarrow 0$
while $i <$ *maximal iterations* **do**
  $X' \leftarrow$ Generate adversarial features using Eq. (14)
  $h_i \leftarrow$ Compute node similarity using Eq. (15)
  if $H < h_i$ **then**
    $X' \leftarrow$ Update
  end if
  $i \leftarrow i + 1$
end while
Update V', E', and X'
return $G' = (V', E', X')$

where γ_i stands for the community distribution parameter for node i, and ϕ_{ij} refers to the probability that the j_{th} feature is assigned to the community. This process achieves optimal parameter learning by minimizing the variational free energy function:

$$\mathcal{T} = \mathbb{E}_q[\log p(X, \theta, k|\alpha, \beta)] - \mathbb{E}_q[\log q(\theta, z)]. \tag{8}$$

After convergence, the community distribution for node i is derived by normalizing γ_i as follows:

$$\theta_i = \frac{\exp(\gamma_i)}{\sum_{k=1}^{K} \exp(\gamma_{ik})}. \tag{9}$$

Based on the community distribution θ_i of each node, we construct its overlapping community partition. Specifically, when the probability of a node belonging to a certain community exceeds the community affinity parameter ζ, the node is considered to fall under that community:

$$\mathcal{C} = \begin{cases} 1 & \text{if } \theta_i \geq \zeta, i \in \{1, 2, \ldots, k\} \\ 0 & otherwise \end{cases}. \tag{10}$$

The resulting candidate set $\mathcal{C}$ functions as a structure-aware prior for the subsequent design of perturbation strategies, providing a foundation for generating high-performance perturbations.

TABLE II
KEY CHARACTERISTICS OF FIVE REAL DATASETS

Datasets	Nodes	Edges	Features	Densities
Citeseer	2110	3668	3703	0.00165
Cora	2485	5069	1433	0.00164
Cora-ML	2810	7981	2879	0.00202
BlogCatalog	5196	171743	8189	0.12724
Flickr	89250	449878	500	0.00011

C. Structure-Aware Link Generation

We analyzed various connection strategies for perturbed data to assess how nodes of varying importance levels within the candidate set influence GNN performance. Based on node importance, the feature connections for fake nodes in SNIA were divided into three strategies: random connection (SNIA-R), maximum degree connection (SNIA-D), and maximum betweenness connection (SNIA-B). Each attack strategy selects nodes with different levels of importance from the candidate set to examine how different perturbation generation methods affect GNN classification outcomes.

In SNIA-R, the model randomly selects $\triangle E$ nodes from the candidate set to connect with the fake nodes. This connection strategy evaluates how nodes of varying levels of importance within the candidate set affect the decision-making process of the classifier, thereby assessing the robustness of the attack. In SNIA-D, the model prioritizes choosing $\triangle E$ nodes with the highest degree from the candidate set to establish fake node connections. The node degree is calculated as follows:

$$D_i = \sum_{j=1}^{n} \mathbb{1}(A_{ij}), \tag{11}$$

where $\mathbb{1}$ indicates the connection between the ith node and its neighboring nodes in the adjacency matrix A. This strategy assesses the impact of the most locally significant nodes within the candidate set on the classifier's performance. In SNIA-B, the model selects $\triangle E$ nodes with the highest betweenness centrality from the candidate set to establish fake node connections. The betweenness centrality of a node is determined as follows:

$$BC_i = \sum_{s \neq i \neq t} \frac{\sigma_{st}(i)}{\sigma_{st}}, \tag{12}$$

where σ_{st} signifies the number of shortest paths between nodes s and t, and $\sigma_{st}(i)$ implies the number of those paths passing through node i. This connection strategy evaluates the impact of globally important nodes within the candidate set on the classifier's performance.

D. Imperceptible Feature Generation

SNIA seeks to introduce subtle data perturbations that remain undetected by GNNs during training. Consequently,

TABLE III
BASELINE MODEL AND ITS RESPECTIVE ADJUSTMENTS IN EXPERIMENTAL VALIDATION

No.	Attack	Year	Target	Modification	Remark
1	Nettack	2018	Graph modification attack	Global injection attack	Disturbance sampling
2	AFGSM	2020	Target injection attack	Global injection attack	Global extension
3	G-NIA	2021	Target injection attack	Global injection attack	Global extension
4	GANI	2024	Global injection attack	—	No modification
5	LPGIA	2024	Global injection attack	—	No modification
6	SNIA-R	2024	Global injection attack	—	Our strategy
7	SNIA-D	2024	Global injection attack	—	Our strategy
8	SNIA-B	2024	Global injection attack	—	Our strategy

the features of injected fake nodes must closely match those of existing nodes. To achieve consistency across various node feature types, SNIA normalizes these features, with the normalization process formulated as follows:

$$X = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{13}$$

where X_{max} and X_{min} are the maximum and minimum values in the feature matrix X, respectively. As expressed in Eq. (2), the GNN aggregation scheme clearly operates within the two-hop neighborhoods. Consequently, SNIA restricts the feature similarity to the two-hop neighbors of the target node, ensuring consistency between fake and original nodes in the network. The feature generation process for fake nodes is outlined as follows:

$$X^{'}[\Delta F_i] = \frac{\sum_{j=1}^{\mathcal{N}} X[v, \Delta F_i]}{\sum_{j=1}^{\mathcal{N}} \mathbb{1}(X[v, \Delta F_i] \neq 0)}, \tag{14}$$

where ΔF represents the feature budget for the node, and $\mathcal{N}$ indicates the set of neighboring nodes of the target node.

According to Eq. (14), the SNIA method generates fake node features by allocating the feature attack budget ΔF. It iteratively selects feature values from the corresponding positions in neighboring nodes until the entire budget is exhausted. The resulting attributes of the fake nodes are derived as the mean values of the selected positions. For discrete features, the average feature value is consistently 1, whereas for continuous features, the value remains continuous after averaging. This method allows the SNIA scheme to effectively generate features that accommodate both discrete and continuous types, catering to a wide array of real-world applications.

Earlier studies have demonstrated that the adaptability of node injection attack techniques can, to a certain degree, disrupt the structural characteristics of the network [23]. In this study, we introduce a limitation on the homogeneity between the generated fake nodes and their neighboring nodes, thereby producing perturbations that are harder to identify. This limitation mandates that the fake nodes exhibit significant homogeneity with their neighbors. The approach for computing the homogeneity constraint is as follows:

$$h_i = cos\left(r_i, X_i'\right), \tag{15}$$

where cos denotes the cosine similarity, X_i' indicates the generated feature of fake node, and r_i is defined as follows:

$$r_i = \sum_{j=1}^{\mathcal{N}} \frac{1}{\sqrt{d_i}\sqrt{d_j}} X_j, \tag{16}$$

where d_i and d_j are the degree of the node i and j, respectively. This generation constraint enhances the feature generation process, ensuring local coherence between the fake node features and the original network feature.

E. Time Complexity Analysis

This section presents the pseudocode for the SNIA scheme in Algorithm 1 and examines its time complexity to highlight its effectiveness. The SNIA method comprises two key components: node connection perturbation and fake node feature generation. Specifically, the computational cost for generating perturbation edges is $O(ndk)$, where n indicates the number of network nodes, d denotes the feature dimension, and k signifies the number of planned community divisions. The computational cost for feature generation is $O(nd)$, where n implies the number of neighboring nodes. Consequently, the overall computational complexity of the SNIA algorithm exhibits $O(ndk)$.

Unlike other attack methods that involve training a surrogate model to produce perturbation data before attacking the target model, SNIA bypasses this traditional method, employing a more efficient perturbation generation method. Resultantly, the time complexity of our approach is solely determined by the data generation process, eliminating the need for surrogate model training and ensuring efficient attack execution.

IV. EXPERIMENTS

This section highlights two key components: the experimental setup and experimental analysis. Through comprehensive experimentation, we validate the effectiveness of the proposed approach. The experimental setup includes details on the five datasets, the baseline models, and the target models subjected to attack. The experimental analysis evaluates the performance of the SNIA scheme through comparative experiments, sensitivity analysis of parameters, and ablation studies.

TABLE IV
NODE CLASSIFICATION RESULTS OF THREE STANDARD GNN MODELS ACROSS FIVE DATASETS

Datasets	Victim	Clean	Nettack	AFGSM	GNIA	GANI	LPGIA	SNIA-R	SNIA-D	SNIA-B
Citeseer	GCN	0.7447	0.7198	0.7476	0.7286	0.7156	0.7306	**0.7049**	0.7110	0.7085
	GAT	0.7417	0.7314	0.7402	0.7393	0.7328	0.7379	0.7037	0.7014	**0.6955**
	SGC	0.7441	0.7320	0.7494	**0.7204**	0.7470	0.7350	0.7287	0.7251	0.7268
Cora	GCN	0.8490	0.8305	0.8375	0.8053	0.8410	0.8387	**0.8003**	0.8014	0.8048
	GAT	0.8319	0.8395	0.8342	0.8265	0.8224	0.8378	0.7927	**0.7801**	0.7947
	SGC	0.8314	0.8290	0.8314	0.8133	0.8134	0.8282	0.8129	**0.8089**	0.8133
Cora-ML	GCN	0.8581	0.8479	0.8456	0.8425	0.8431	0.8541	0.8274	**0.8105**	0.8193
	GAT	0.8536	0.8421	0.8350	0.8190	0.8296	0.8440	0.8078	**0.7945**	0.8033
	SGC	0.8372	0.8280	0.8324	0.8243	0.8211	0.8367	**0.8185**	0.8211	0.8207
BlogCatalog	GCN	0.7670	0.7589	0.7654	0.7738	0.7479	0.7596	0.7168	0.7224	**0.7139**
	GAT	0.6579	0.6233	0.6263	0.6310	0.6259	0.6365	0.5711	**0.5708**	0.5823
	SGC	0.7426	0.7410	0.7420	0.7410	0.7412	0.7411	0.7383	0.7380	**0.7375**
Flickr	GCN	0.4740	0.4749	0.4735	0.4452	0.4719	0.4653	0.4279	0.4275	**0.4268**
	GAT	0.4710	0.4698	0.4644	0.4551	0.4692	0.4661	0.4226	**0.4217**	0.4226
	SGC	0.4342	0.4226	0.4273	0.4230	0.4236	0.4226	**0.4197**	0.4211	0.4208

A. Experimental Setup

1) Datasets: We considered five representative real-world network datasets for our experiments: Citeseer, Cora, Cora-ML, BlogCatalog, and Flickr. These datasets were selected to cover various network scales, data types, and densities, encompassing diverse practical application scenarios. To mitigate the influence of independent communities or isolated nodes on the experimental results, we followed the experimental setup from prior studies, concentrating on the largest connected component of the network. The relevant metrics for these specific datasets are provided in Table II.

2) Victim Models: To evaluate the computational performance of the proposed model on various victim GNNs, we selected two prominent sets of target models widely recognized in the field of adversarial attacks. The first group consists of key GNNs, namely GCN [26], graph attention network (GAT) [30], and simplified graph convolution (SGC) [31]. The second group comprises defensive GNNs, specifically the node similarity preserving model (SimpGCN) [32] and the perturbed edge detection model (GNNGuard) [33], developed to counteract the effects of adversarial data perturbations on GNNs prediction accuracy. These two GNN groups were chosen to represent practical scenarios encountered by attack strategies.

3) Baseline Models: To validate the effectiveness of the proposed method, we compared the SNIA model with five well-established attack methods, divided into three categories. The first category includes graph modification attacks, exemplified by network attack (Nettack) [12]. The second category consists of target node attacks, represented by adversarial fast gradient sign method (AFGSM) [19] and generalizable node injection attack (G-NIA) [20]. The final category includes global node attacks, represented by global attack node injection (GANI) [21] and label-propagation-based global injection attack (LPGIA) [34]. The selected baseline model comprehensively incorporates recent graph adversarial attack strategies to examine and compare the variations among these methods. Since various strategies target distinct objectives, we adjusted some modifications to suit global injection attacks. These modifications are outlined in Table III.

4) Parameter Settings: To ensure consistency in our experimental results, we adhered to the settings established in prior studies regarding dataset split ratios and edge attack budgets [34]. Specifically, the dataset was partitioned into 10% for training, 10% for validation, and 80% for testing. The edge attack budget is determined based on the network density: if the density is below 0.1, we set the edge budget to the average node degree, as per [34]. To minimize attention from the GNN model in dense networks (i.e., when the density exceeds 0.1), the edge budget is set to the minimum node degree within the candidate node set. Moreover, since altering all feature dimensions of a node is impractical in real-world scenarios, we adopted the approach in [21] for the node feature attack budget, which is defined as the average number of non-zero features per node in the network. The number of communities k is set equal to the number of node label classes, and the community affinity parameter ζ is fixed at 0.4. In our validation experiments, the node attack budget for all algorithms is uniformly set to 1, while other parameters follow the default configurations of each respective method.

B. SNIA Performance Under Regular GNNs

This section evaluates the performance of all baseline models across three representative regular GNN architectures using five datasets. The detailed experimental results are shown in Table IV. These findings demonstrate that SNIA outperforms all other attack strategies, except for the SGC model on the Citeseer dataset, where it slightly lags by 0.47% behind G-NIA. This minor difference is due to the advantages of the white-box attack executed by G-NIA.

All three SNIA-based attack strategies exhibit significant perturbation effects, with a minimal gap between them in an overall perturbation of up to 8.71%. Although the optimal

TABLE V
NODE CLASSIFICATION RESULTS OF TWO DEFENSIVE GNN MODELS ACROSS FIVE DATASETS

	Citeseer		Cora		Cora-ml		BlogCatalog		Flickr	
	SimpGCN	GNNGuard	SimpGCN	GNNGuard	SimpGCN	GNNGuard	SimpGCN	GNNGuard	SimpGCN	GNNGuard
Clean	0.7571	0.7292	0.8244	0.8189	0.8576	0.8563	0.9165	0.7643	0.4671	0.4774
Nettack	0.7351	0.7233	0.8284	0.8189	0.8381	0.8416	0.8703	0.7333	0.4429	0.4696
AFGSM	0.7476	0.7287	0.8199	0.8355	0.8501	0.8389	0.8703	0.7604	0.4369	0.4648
G-NIA	0.7393	0.7233	0.8269	0.8104	0.8505	0.8629	0.9014	0.7537	0.4596	0.4656
GANI	0.7322	0.7310	0.8013	0.8085	0.8416	0.8438	0.8891	0.7623	0.4342	0.4687
LPGIA	0.7452	0.7254	0.8283	0.8349	0.8509	0.8521	0.9003	0.7575	0.4601	0.4703
SNIA-R	0.7150	0.7085	0.7927	**0.7928**	0.8194	**0.8133**	**0.8433**	0.7241	**0.4221**	**04225**
SNIA-D	**0.7043**	0.7126	**0.7902**	0.8003	**0.8185**	0.8233	0.8573	**0.7171**	0.4240	0.4297
SNIA-B	0.7174	**0.7079**	0.7932	0.8048	0.8278	0.8238	0.8545	0.7244	0.4226	0.4252

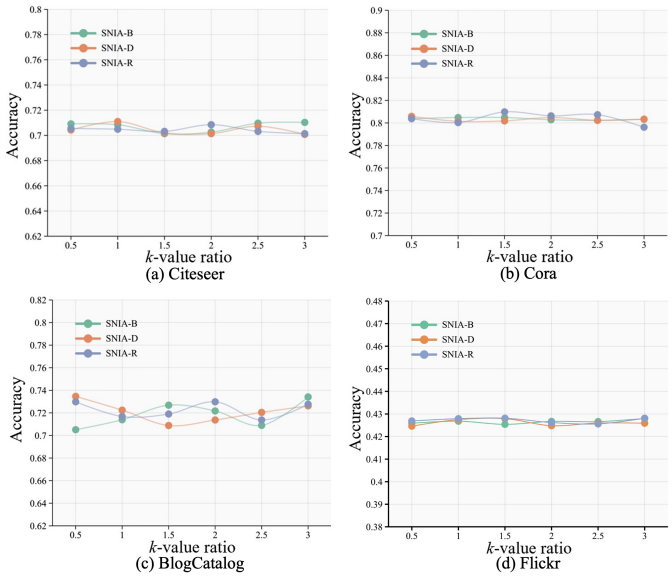

Fig. 3. Analysis of parameter sensitivity in the SNIA approach.

perturbation effect varies across different strategies, each one outperforms the latest attack method. Furthermore, SNIA strategies consistently produce stable perturbations across all models, despite differences in how GNNs process data. Experimental results on regular GNNs validate the robust generalization capability of the SNIA scheme.

C. SNIA Performance Under Defensive GNNs

To demonstrate the robustness of SNIA and the generalization capability of its attack strategies, we evaluated two prominent defense-oriented GNNs. The detailed experimental results are presented in Table V. In experiments with defensive GNNs, SNIA along with its attack methods displayed consistent attack performance, indicating that, under identical data and model conditions, the effectiveness of the attacks remains highly stable across different attack methods.

Similarly to the results observed on regular GNNs, the optimal perturbation effects are observed across SNIA-R, SNIA-D, and SNIA-B, respectively, with each attack strategy performing exceptionally well on various datasets. Even in datasets and defensive models where other attack strategies falter, SNIA achieves a perturbation effect of 7.32%, reinforcing its robust generalization capability. Coupled with the experimental results outlined in Table IV, the proposed SNIA scheme demonstrates strong attack effectiveness against both regular and defensive GNNs, highlighting its ability to bypass defense mechanisms.

D. Ablation Studies

This section presents ablation studies to evaluate the contribution of different mechanisms within SNIA. The SNIA model consists of two key modules: structure-aware node selection for edge generation and imperceptible neighbor similarity perturbation for feature generation. We remove one or both mechanisms to analyze the impact of each module on the attack's performance. The SNIA_{l-f} variant removes all modules, generating perturbation edges and features randomly. SNIA_{l} omits only the edge generation module, selecting edge connections randomly from the entire network. In contrast, SNIA_{f} eliminates the feature generation module, using a random feature generation approach.

The experimental outcomes of the ablation study are presented in Table VI. In the SNIA attack scheme, the feature generation module offers a more subtle and effective attack than the edge generation module; however, the absence of global perturbation reduces the visibility of the attack. The edge generation strategy ensures a reliable attack effect. Nevertheless, without constraining the features of fake nodes, the perturbed features appear too abrupt. As a result, neither module achieves a powerful attack effect. In contrast, the combined use of both attack modules demonstrates a robust attack across all datasets.

E. Parameter Sensitivity Analysis

To assess the effect of community count on the SNIA attack, we performed a parameter sensitivity analysis, examining how different values of the community parameter k impact the actual effects of the attack. In this section, we used the base community number as the starting point, varying other community values between 0.5 and 3 in increments of 0.5. Since the Cora and Cora-ML datasets are derived from different processing versions of the same raw dataset and yield

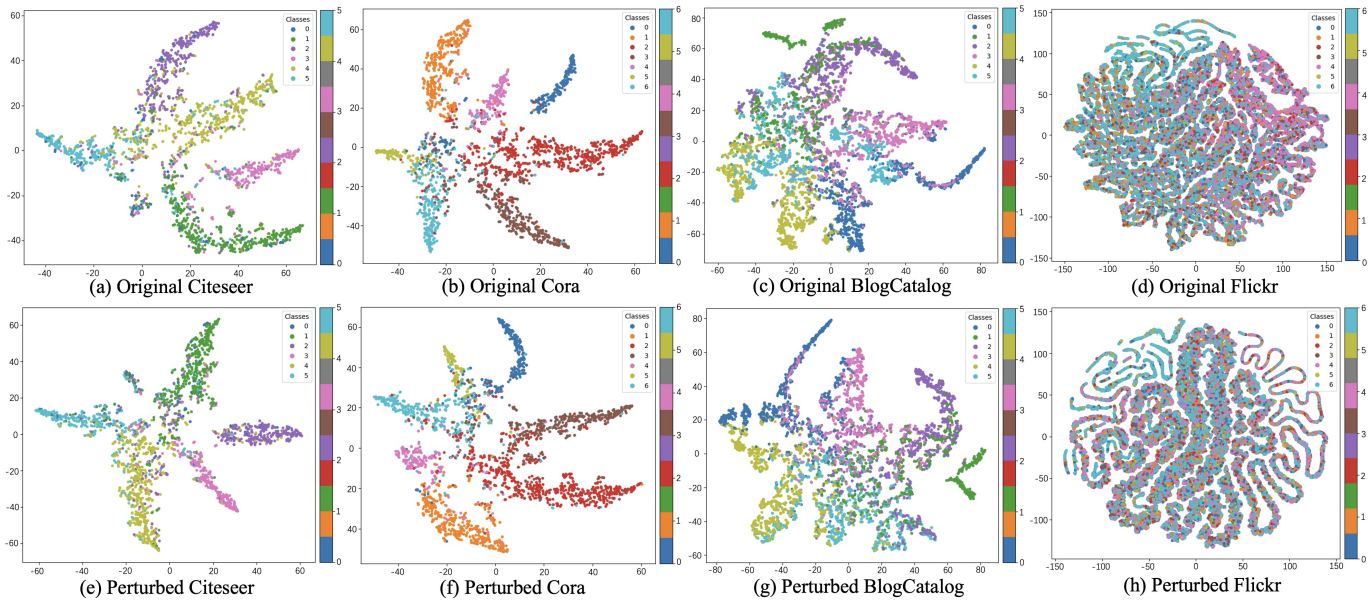

Fig. 4. Visualization results of the GNN-based node classification predictions before and after the implementation of the SNIA attack strategy.

TABLE VI
OUTCOMES OF ABLATION EXPERIMENTS USING THE SNIA APPROACH

Modules	Citeseer	Cora	Cora-ML	BlogCatalog	Flickr
Clean	0.7571	0.8244	0.8576	0.9165	0.4671
SNIA$_{l-f}$	0.7440	0.8184	0.8514	0.8992	0.4597
SNIA$_{l}$	0.7268	0.7997	0.8425	0.8794	0.4525
SNIA$_{f}$	0.7352	0.7952	0.8394	0.8621	0.4502
SNIA	**0.7043**	**0.7902**	**0.8185**	**0.8573**	**0.4240**

consistent experimental results, we present only the results for the Citeseer, Cora, BlogCatalog, and Flickr datasets in Fig. 3. These results illustrate the performance of SNIA under diverse parameter settings.

The experimental findings indicate that all three attack strategies of SNIA consistently maintain their effectiveness across all datasets. The perturbation results remain stable and are unaffected by variations in the number of communities k. This further confirms the robustness of the SNIA method, which continues to deliver effective attacks despite changes in network structure or community assignments. Furthermore, this highlights the adaptability of the SNIA approach, stressing its relevance in real-world scenarios.

F. Visualization Studies

In this section, we utilize the t-SNE algorithm [35] to visualize data representations post-model training, aiming to assess the impact of SNIA on GNN classification performance before and after the attack. The datasets selected align with those used in the parameter sensitivity experiments. Figure 4 presents the t-SNE visualization results of SNIA across four datasets: Figs. 4(a)–(d) depict the training outcomes of clean GNN models on the Citeseer, Cora, BlogCatalog, and Flickr datasets, while Figs. 4(e)–(h) display the corresponding results after applying SNIA perturbations on the same datasets.

The observations from the experiments reveal that on the Citeseer and Cora datasets, GNNs effectively classify nodes in clean graphs, showing clear class boundaries and compact intra-class clustering. However, following the SNIA attack, these class boundaries become significantly blurred, and intra-class clusters are noticeably dispersed. For the BlogCatalog and Flickr datasets, while the initial class boundaries are less distinct, nodes within each class still exhibit some degree of cohesion. After the SNIA attack, classification confusion increases markedly, with a pronounced effect observed in the Flickr dataset. These visualization results illustrate that SNIA severely degrades classification performance across various datasets, demonstrating that even the injection of a single node can compromise the accurate classification of a substantial number of nodes.

G. Graph Distribution Analysis

As follows, we analyze the extent to which different attack strategies perturb the graph structure. Our evaluation is conducted from two perspectives. First, we visualize the homophily distributions of the original and poisoned graphs to intuitively illustrate the differences in perturbation. Second, we quantitatively measure structural changes by computing the KL divergence [36] between the original and poisoned graphs. To better observe the impact, we conducted experiments on the Citeseer dataset with all parameters kept consistent. Figure 5 showcases a thorough graphical depiction of our analysis.

Previous evaluations suggest that stronger manipulations of graph structures can significantly impair the classification performance of GNNs. However, such manipulations are also more susceptible to detection and mitigation by GNN-based defense mechanisms. The distribution results reveal that, irrespective of the specific SNIA variant employed, SNIA introduces the least structural perturbation compared to other state-of-the-art attack methods. This implies that by incorporating structural constraints, SNIA maintains high attack effectiveness while minimizing disruption to the original

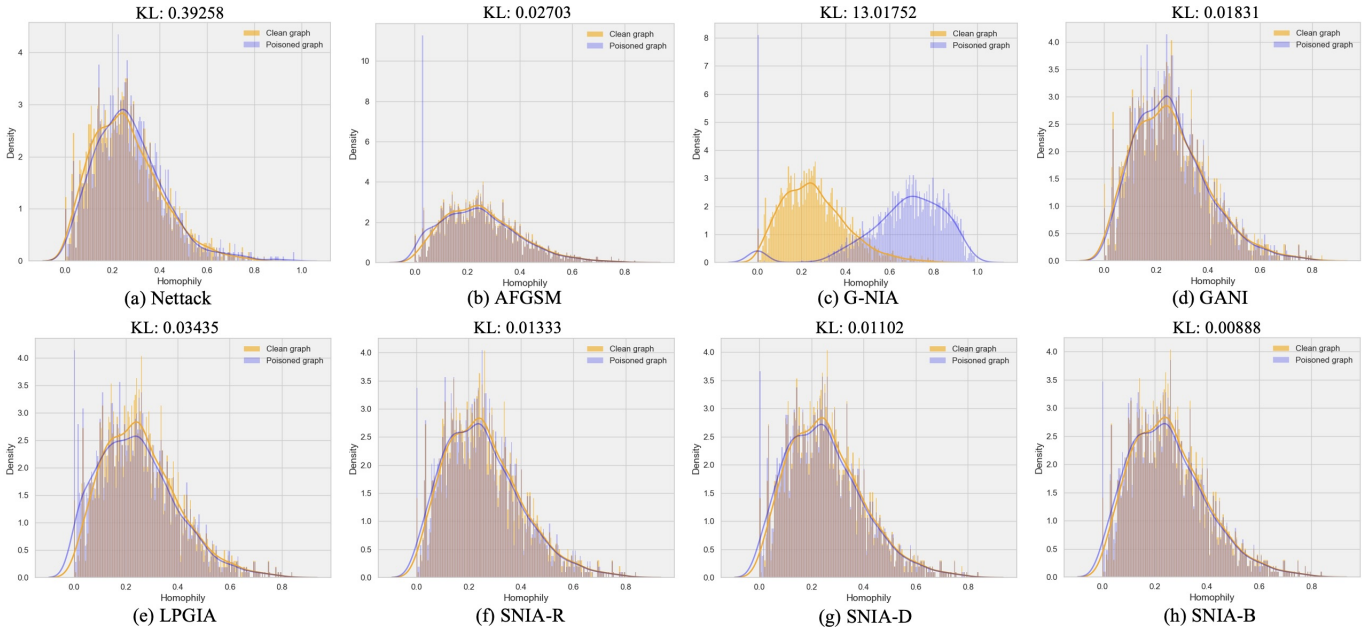

Fig. 5. Homophily distribution under different attack strategies compared to the distribution without attack.

graph structure. These findings further manifest that SNIA's efficacy in disrupting GNNs does not depend on damaging the graph structure; instead, preserving that structure is crucial for evading detection by GNN defenses.

V. Conclusion

This study explores the impact of node injection techniques on the training accuracy of GNNs, with due consideration of network structure. In contrast to prior research, our proposed algorithm achieves a significant reduction in GNN predictive accuracy even under minimal attack conditions. We develop the attack strategy by leveraging network topology and targeting nodes prone to belonging to overlapping communities, thereby maximizing the effectiveness of the injected node. To evade detection by the GNN model, we impose homogeneity constraints during the feature generation process. This ensures that the injected fake node closely resembles its neighbors, enhancing the stealthiness of the attack. Extensive experimental results verify that this method effectively disrupts GNN models, including those equipped with defense mechanisms.

Numerous studies on graph injection attacks have shown that perturbing graph-structured data can considerably compromise the robustness of GNN models. However, a prevalent challenge with node injection attacks is the mismatch between the injected perturbation nodes and the real nodes in the original graph. Such discrepancies may result in high node reconstruction errors, creating potential avenues for detecting and defending against adversarial perturbations. Future research should thus focus on minimizing node reconstruction errors and strengthening the consistency between node features and graph structure, so as to enhance the robustness of GNNs in adversarial settings.

References

[1] S. Ryoma, Y. Makoto, and K. Hisashi, "Constant time graph neural networks," *ACM Transactions on Knowledge Discovery from Data*, vol. 16, no. 5, pp. 1–31, 2022.

[2] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 2005, pp. 729–734.

[3] Y. Liu, S. Zhao, X. Wang, L. Geng, Z. Xiao, and J. C.-W. Lin, "Self-consistent graph neural networks for semi-supervised node classification," *IEEE Transactions on Big Data*, vol. 9, no. 4, pp. 1186–1197, 2023.

[4] S. Vashishth, S. Sanyal, B. Ray, and P. Talukdar, "Composition-based multi-relational graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, 2020.

[5] J. Wang, P. Chen, B. Ma, J. Zhou, Z. Ruan, G. Chen, and Q. Xuan, "Sampling subgraph network with application to graph classification," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 4, pp. 3478–3490, 2021.

[6] L. Sang, M. Xu, S. Qian, and X. Wu, "Adversarial heterogeneous graph neural network for robust recommendation," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 5, pp. 2660–2671, 2023.

[7] B. Pietro, P. Niccolò, B. Asma, S. Franco, M. Marco, and B. Monica, "Composite graph neural networks for molecular property prediction," *International Journal of Molecular Sciences*, vol. 25, no. 12, p. 6583, 2024.

[8] Y. Zhang, N. Yuan, Z. Zhang, J. Du, T. Wang, B. Liu, A. Yang, K. Lv, G. Ma, and B. Lei, "Unsupervised domain selective graph convolutional network forpreoperative prediction of lymph node metastasis in gastric cancer," *Medical Image Analysis*, vol. 79, p. 102467, 2022.

[9] T. T. Nguyen, D. K. Q. Nguyen, T. N. Thanh, T. H. Thanh, H. V. Viet, L. N. Phi, J. Jun, and V. H. N. Quoc, "Poisoning GNN-based recommender systems with generative surrogate-based attacks," *ACM Transactions on Information Systems*, vol. 41, no. 3, pp. 1–24, 2023.

[10] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *Proceedings of the International Conference on Learning Representations*, 2018.

[11] L. Sun, Y. Dou, C. Yang, K. Zhang, J. Wang, P. S. Yu, L. He, and B. Li, "Adversarial attack and defense on graph data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 8, pp. 7693–7711, 2022.

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

[13] B. Wang and N. Z. Gong, "Attacking graph-based classification via manipulating the graph structure," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2023–2040.
[14] X. Lin, C. Zhou, H. Yang, J. Wu, H. Wang, Y. Cao, and B. Wang, "Exploratory adversarial attacks on graph neural networks," in *Proceedings of the IEEE International Conference on Data Mining*, 2020, pp. 1136–1141.
[15] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *Proceedings of the International Conference on Machine Learning*, 2018, pp. 1115–1124.
[16] M. Zhang, L. Hu, C. Shi, and X. Wang, "Adversarial label-flipping attack and defense for graph neural networks," in *Proceedings of the IEEE International Conference on Data Mining*, 2020, pp. 791–800.
[17] Y. Sun, S. Wang, X. Tang, T. Hsieh, and V. G. Honavar, "Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach," in *Proceedings of the Web Conference*, 2020, pp. 673–683.
[18] Y. Chen, Z. Ye, Z. Wang, and H. Zhao, "Imperceptible graph injection attack on graph neural networks," *Complex and Intelligent Systems*, vol. 10, no. 1, pp. 869–883, 2023.
[19] 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.
[20] 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 and Knowledge Management*, 2021, pp. 1794–1803.
[21] J. Fang, H. Wen, J. Wu, Q. Xuan, Z. Zheng, and C. Tse, "GANI: global attacks on graph neural networks via imperceptible node injections," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 4, pp. 5374–5387, 2024.
[22] 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 ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021, pp. 2461–2471.
[23] Y. Chen, H. Yang, Y. Zhang, K. Ma, T. Liu, B. Han, and J. Cheng, "Understanding and improving graph injection attack by promoting unnoticeability," in *Proceedings of the International Conference on Learning Representations*, 2022.
[24] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, pp. 828–841, 2019.
[25] 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.
[26] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, 2017.
[27] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.
[28] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *Proceedings of the IEEE International Conference on Data Mining*, 2013, pp. 1151–1156.
[29] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.
[30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the International Conference on Learning Representations*, 2018.
[31] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the International Conference on Machine Learning*, 2019, pp. 6861–6871.
[32] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, "Node similarity preserving graph convolutional networks," in *Proceedings of the ACM International Conference on Web Search and Data Mining*, 2021, pp. 148–156.
[33] X. Zhang and M. Zitnik, "GNNGuard: Defending graph neural networks against adversarial attacks," *Neural Information Processing Systems*, vol. 33, pp. 9263–9275, 2020.
[34] 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.
[35] L. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 2605, pp. 2579–2605, 2008.
[36] J. M. Joyce, "Kullback-leibler divergence," *Springer Berlin Heidelberg*, 2011.

Fei Yan is a professor at the School of Computer Science and Technology, Changchun University of Science and Technology, China. He is now a visiting professor at the Department of Computer Science and Technology, Tsinghua University, China. He received his doctorate in engineering and worked as a postdoctoral fellow afterward in the Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology, Japan. He is serving as an Associate Editor of the Information Sciences journal and Journal of Advanced Computational Intelligence and Intelligent Informatics. He has published more than 100 papers in the fields of computational intelligence, quantum information processing, machine learning, and health informatics.

Yanlong Tang graduated from the School of Computer Science, Qinghai Normal University, China, with a Master's degree. He is currently a research assistant at the School of Computer Science and Technology, Changchun University of Science and Technology, China. His research interests include natural language processing, representation learning, and graph neural networks.

Witold Pedrycz (F'98) is a professor in the Department of Electrical and Computer Engineering, University of Alberta, Canada. He was elected a foreign member of the Polish Academy of Sciences and a Fellow of the Royal Society of Canada. He received a prestigious Norbert Wiener award from the IEEE Systems, Man, and Cybernetics Society. He is a recipient of the IEEE Canada Computer Engineering Medal, a Cajastur Prize for Soft Computing from the European Centre for Soft Computing, a Killam Prize, and a Fuzzy Pioneer Award from the IEEE Computational Intelligence Society. He is the Editor-in-Chief of Information Sciences (*Elsevier*), WIREs Data Mining and Knowledge Discovery (*Wiley*), and Journal of Data, Information and Management (*Springer*), etc. He has total citations of more than 110,000, an h-index of 139, and an i10-index of 1,472 based on Google Scholar.

Kaoru Hirota (LM'16) is an emeritus professor at the Tokyo Institute of Technology in Japan and a distinguished professor at the School of Automation, Beijing Institute of Technology, China. From 2015 to 2021, he served as the director of the Beijing representative office of the Japan Society for the Promotion of Science (JSPS). He is a fellow and former president of the International Fuzzy Systems Association (IFSA) and has also served as the president of the Japan Society for Fuzzy Theory and Systems (SOFT). He has received several prestigious awards, including the Banki Donat Medal, the Henri Coanda Medal, the Chinese Friendship Medal, and the Grigore MOISIL Award. He is currently the Editor-in-Chief of the Journal of Advanced Computational Intelligence and Intelligent Informatics, as well as an advisory editor and associate editor for several prestigious international journals. He has organized numerous international conferences and workshops as a Founding and General Chair.

Structure-Aware Attack on Graph Neural Networks via Imperceptible Node Injection

Fei Yan, Yanlong Tang, Witold Pedrycz, Kaoru Hirota

***Abstract*—Graph neural networks (GNNs) excel in various graph-based tasks due to their exceptional ability to process non-Euclidean data. However, recent research indicates that their predictive performance is highly vulnerable to perturbations from intentionally manipulated data. Current node injection attack methods disrupt GNN training by injecting numerous nodes, often in excessive amounts, making them easily detectable. To address this issue, this study introduces a structure-aware node injection attack (SNIA), which enables effective and subtle attacks under extreme budget constraints. The scheme leverages the network topology to construct an attack candidate set and applies homogeneity constraints to regulate the generation of perturbed features. By eliminating the dependency on surrogate models for generating perturbed data, SNIA effectively diminishes the global classification performance of GNNs based on the network's inherent structure. We conducted attack experiments with the SNIA scheme on real-world network datasets against both general and defensive GNNs. The experimental findings reveal that it significantly surpasses the existing state-of-the-art methods in attack efficiency, while also showcasing outstanding generalization and resilience.**

***Index Terms*—Graph neural networks (GNNs), imperceptible attack, node injection, homophily constraint.**

I. Introduction

GRAPHS serve as powerful abstract mathematical models, effectively capturing complex relationships among entities in the real world [1]. Their non-Euclidean structure makes processing data inherently challenging. Graph neural networks (GNNs) address this challenge by enabling the processing of non-Euclidean data [2], uncovering hidden insights regarding graph data, and applying them effectively to tasks such as node classification [3], link prediction [4], and graph classification [5]. Owing to their computational efficiency and precision, GNNs are increasingly used in fields such as recommendation systems [6], bioinformatics [7], and medical diagnosis support [8]. However, recent studies highlight that GNNs are susceptible to adversarial attacks [9], which can compromise their robustness [10] and exhibit serious implications in areas such as genomic analysis and auxiliary diagnosis in medicine.

Initial perturbations primarily focus on altering the original graph structure to mislead GNNs [11]. These attacks generally manipulate node features and graph topology using gradient-based methods [12], approximate optimization [13], and adversarial exploration attacks [14] to modify graph data. Additionally, some perturbation strategies employ reinforcement learning to invert the node connectivity relationship [15]. Beyond adversarial perturbations that alter graph structures and node features, some approaches focus on modifying true labels through gradient optimization to achieve desired perturbations [16]. However, these methods often lack practical applicability. As an example, in citation networks, attackers face challenges in altering existing citation relationships between articles. Furthermore, modifying the original data requires access to the underlying databases, necessitating significant operational authority from the attacker [17], which limits the feasibility of these techniques.

Node injection attack methods overcome the limitations of traditional graph perturbation methods in real-world applications. These attacks create perturbation data based on the original graph structure, eliminating the need to alter the original graph structure. For instance, generating new profile pages on social networks can form new node relationships without modifying the original data, thereby minimizing the need for elevated operational permissions [18]. Attackers can generate perturbations by using the gradient of the target node loss function [19] and optimization methods [20], similar to altering graph structures. In addition, some studies have proposed poisoning attacks using genetic algorithms [21], Q-learning [17], and topological vulnerabilities [22]. These methods typically involve injecting a certain number of nodes to disrupt GNN training. However, excessive node injection can waste attack resources, and large-scale injections increase the risk of detection by GNN models. Current research indicates that excessive node injection attacks can disrupt graph homophily, enabling robust GNNs to detect structural anomalies and apply appropriate defenses [23]. Extreme cases of such attacks are initially explored in computer vision, where a single pixel injection into an image can misclassify the model [24]. Some studies have explored extreme injection strategies to manipulate the classification of target nodes during GNN training [20], [25]. However, research on reducing global node classification performance in such scenarios is scarce. Additionally, most node injection attack methods rely on building a surrogate model and generating perturbations

This work was supported by the Natural Sciences and Engineering Research Council of Canada (Grant Number: RGPIN-2022-03045).

Fei Yan is with School of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130022, China and Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: yanfei@cust.edu.cn). Corresponding author: Fei Yan.

Yanlong Tang is with School of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130022, China (e-mail: supertyl@outlook.com).

Witold Pedrycz is with Department of Electrical and Computer Engineering, University of Alberta, Edmonton T6G 1H9, Canada (e-mail: wpedrycz@ualberta.ca).

Kaoru Hirota is with School of Computing, Tokyo Institute of Technology, Yokohama 226-8502, Japan and School of Automation, Beijing Institute of Technology, Beijing 100081, China (e-mail: hirota@bit.edu.cn).

using gradient information and optimization processes. These methods necessitate training a separate GNN model, which is time-consuming for practical attacks.

This study examines node injection attack techniques that significantly influence the global prediction outcomes of GNNs under minimal attack resources. We introduce a novel data perturbation approach, named structure-aware node injection attack (SNIA), which leverages the inherent characteristics and structural coherence of graph data to substantially degrade the global classification performance of the target model. Extensive experimental results validate that the proposed SNIA demonstrates high generalizability and resilience, successfully bypassing detection by defensive models. The key contributions of this study are summarized as follows:

1) We propose an imperceptible attack scheme that disrupts GNN training under extreme budget constraints by injecting individual nodes into the original network data, ultimately reducing their overall classification performance.
2) To optimize the effectiveness of the attack, we depart from the conventional approach of using surrogate models for data perturbation. Instead, we employ a macro-level strategy that emphasizes overlapping community assignments for network nodes to create a candidate attack set.
3) To enhance the stealth of this method, we restrict the generated perturbations to maintain strong homophily with the two-hop neighbors of target nodes, thereby minimizing the impact on the original network structure.

The remainder of this paper is organized as follows. Section II reviews GNNs and related research on graph injection attacks. Section III presents a comprehensive discussion of the proposed attack scheme, covering node community affiliation, overlapping community detection, and subtle feature generation. Section IV outlines a series of experiments designed to validate the effectiveness of the proposed method and provides the necessary parameter settings for the reproducibility of the experiments. Finally, Section V concludes the key findings of this study and discusses future directions for the research.

II. Preliminaries

This section defines the key concepts related to GNNs and node injection attacks to clarify the SNIA strategy proposed in this study. Table I lists the standard symbolic notations.

A. Graph Neural Networks (GNNs)

Given an attribute graph $G = (V, E, X, Y)$, where $V \in \{v_1, v_2, \ldots, v_n\}$ implies the set of graph nodes; $E = \{e_{ij} \mid 0 < i, j \leq n\}$ refers to the set of edges; e_{ij} signifies an edge between nodes i and j; $X \in \mathbb{R}^{n \times d}$ denotes the d-dimensional feature matrix of the graph; Y indicates the label set of graph nodes; and $n = |V|$ represents the total number of nodes in the graph. The linking relationships between nodes are described by the adjacency matrix A, where $A_{i,j} = 1$ if an edge exists between nodes i and j, and $A_{i,j} = 0$ otherwise.

In GNN research, the Graph Convolutional Network (GCN) [26] is the most prominent model, commonly featuring a two-layer graph convolutional architecture. The process of updating node representations in the first layer is performed as follows:

TABLE I
Common Notations and Their Descriptions in SNIA

Notations	Descriptions	Notations	Descriptions
G	Original attribute graph	G'	Poisoned attribute graph
V	Node set of G	V'	Node set of G'
E	Edge set of G	E'	Edge set of G'
A	Adjacency matrix of G	A'	Adjacency matrix of G'
X	Feature matrix of G	X'	Feature matrix of G'
$\mathcal{F}_\theta(\cdot)$	GNN classifier	$\triangle$	Total attack budget
$\triangle F$	Feature attack budget	$\triangle E$	Edge attack budget
y	Real node label	$\tilde{y}$	Predicted node label
v	Node of G	p	Injected fake node

$$H^{(l+1)} = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right), \tag{1}$$

where σ signifies the nonlinear activation function ReLU; $H^{(l)}$ refers to the node representation at the l-th layer; $\hat{A} = \tilde{D}^{-\frac{1}{2}}(A + I)\tilde{D}^{-\frac{1}{2}}$ indicates the normalized adjacency matrix; I is the identity matrix; $\tilde{D}$ denotes the degree matrix; and $W^{(l)}$ represents the learnable parameter weight matrix at the l-th layer. After two GCN update layers, the probability distribution of node category is derived as follows:

$$\mathcal{F}_\theta(A, X) = \text{softmax}\left(\hat{A}\sigma(\hat{A}XW^{(0)})W^{(1)}\right), \tag{2}$$

where $\mathcal{F}_\theta(\cdot)$ implies the node classifier. This study seeks to optimize the classification accuracy on the perturbation graph G' using meticulously crafted perturbation data.

B. Node Injection Attacks

In the node injection attack technique, attackers typically inject a set of artificial fake nodes $P = \{p_1, p_2, \ldots, p_m\}$ into the original network, where m denotes the total number of injected nodes. After executing node injection, the perturbed network $G' = (V', E', X')$ is formed. The adjacency matrix of the perturbed graph is constructed by merging the original node connections with those of the perturbed nodes, while the feature matrix is derived by concatenating the features of the perturbed nodes with the original network features. Consequently, the adjacency matrix A' and feature matrix X' of the perturbed graph are specified as follows:

$$A' = \begin{bmatrix} A & A_p \\ A_p^T & B \end{bmatrix}, X' = \begin{bmatrix} X \\ X_p \end{bmatrix}, \tag{3}$$

where A_p signifies the connections between the injection and original network nodes, B indicates the connection relationships among the injection nodes, and $X_p \in \mathbb{R}^{m \times d}$ denotes the feature matrix of the injected nodes.

C. Problem Definition

This study examines node classification tasks, focusing on the impact of minor perturbations on GNN predictive accuracy. Attackers typically inject a series of perturbations into the

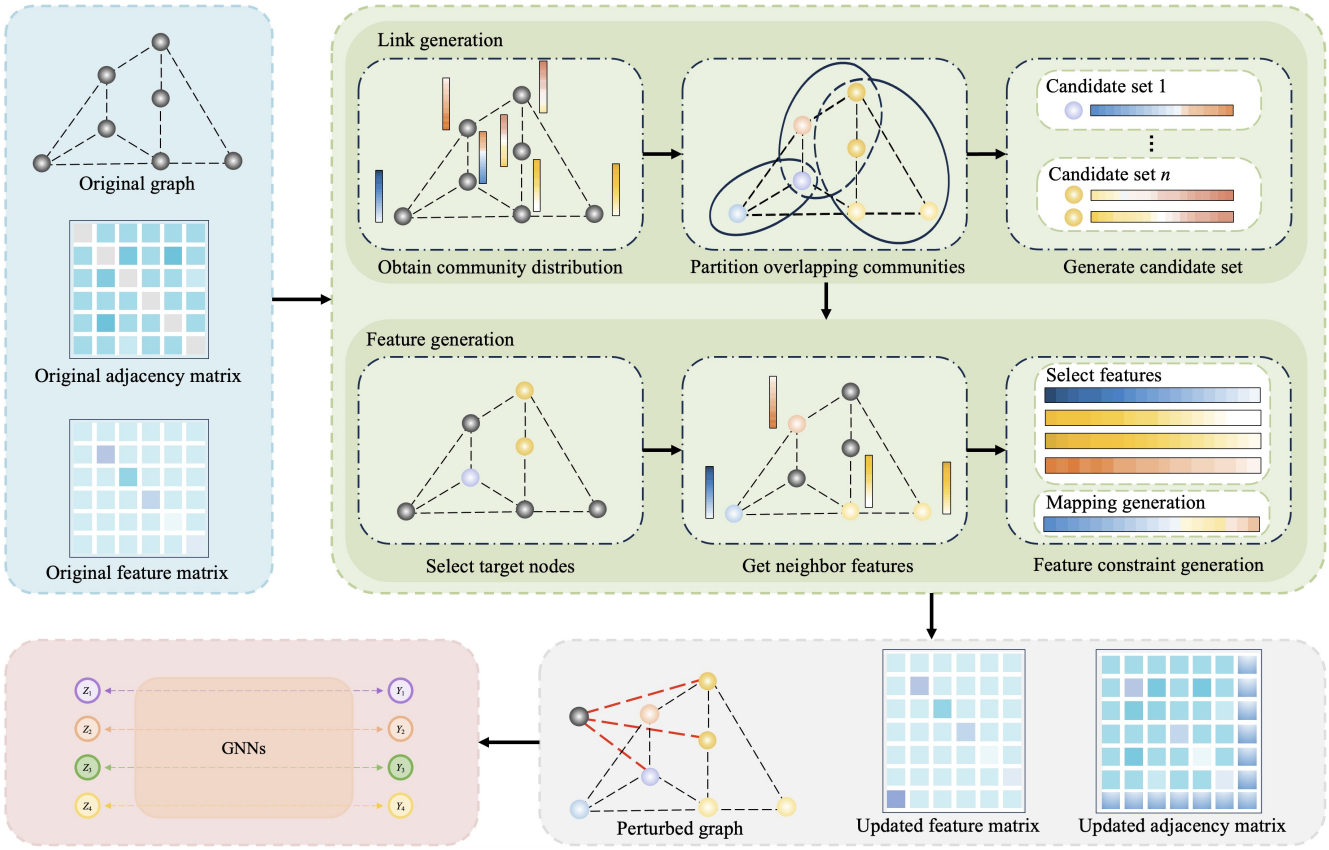

Fig. 1. Framework of the SNIA model. The process consists of three main stages: (1) constructing a set of candidate nodes using the LDA method; (2) selecting target nodes based on a given attack strategy and constructing connections for the poisoned node; and (3) generating features for the poisoned node and injecting a fake node into the original graph to achieve structure-aware perturbation.

original graph to disrupt the training of the target GNN. The task aims to develop a method for generating perturbations, maximizing the degradation of the model's training accuracy. Accordingly, this problem is formally defined as follows:

$$\begin{aligned} &\max \sum_{v=1}^{i} \mathbb{I}\left(\mathcal{F}_{\theta^*}\left(A', X'\right) \neq y_v\right), i \in V_{test} \\ &\text{s.t. } \theta^* = \arg\min_{\theta} \mathcal{L}_{\text{train}}\left(\mathcal{F}_{\theta}\left(A', X'\right)\right), G' - G \leq \Delta \end{aligned}, \quad (4)$$

where V_{test} refers to the test node set; i is the number of nodes in the test set; y_v denotes the true label of node v; $\mathbb{I}$ implies an indicator function that returns 1 if the argument is true and 0 if false; Δ indicates the total attack budget from both feature and link budgets assigned to the injected nodes, measuring the overall changes from the original graph G to the perturbed graph G'; and $\mathcal{L}_{\text{train}}$ represents the training loss, commonly formulated as the cross-entropy function:

$$\mathcal{L}_{\text{train}}(\mathcal{F}_{\theta}(A', X')) = \sum_{v=1}^{j} -y_v \log \tilde{y}_v, j \in V_{train}, \quad (5)$$

where V_{train} signifies the training set of nodes, and j is the number of nodes in the training set. To utilize minor data perturbations and affect the overall prediction accuracy of GNNs, as expressed in Eq. (4), we conducted a comprehensive analysis of the network's structure. The specific methods and implementation details are discussed in Section III.

III. Methodology

This section presents a comprehensive introduction to our proposed node injection attack model, which constructs efficient perturbations by leveraging the intrinsic structural properties of the graph. The scheme encompasses three pivotal components: affinity-based community assignment of nodes (cf. Section III-B), a connection generation strategy for injected nodes (cf. Section III-C), and a stealthy feature construction method (cf. Section III-D).

A. Overall Framework of the SNIA

To effectively illustrate the workflow of SNIA, we present its overall architecture in Fig. 1. A group of densely connected nodes is considered a community, typically composed of entities with similar attributes or types, as noted in [27]. Distinct communities generally exhibit apparent structural separation. The primary objective of GNNs is to classify entities within the same community into a single category. However, in real-world networks, an individual entity may simultaneously belong to multiple communities, a structural property known as overlapping communities [28]. Nodes with such multicommunity tendencies can significantly influence the information propagation process in GNNs due to their involvement in multiple community contexts.

Drawing from this observation, we propose a node injection attack method capable of adequately undermining the

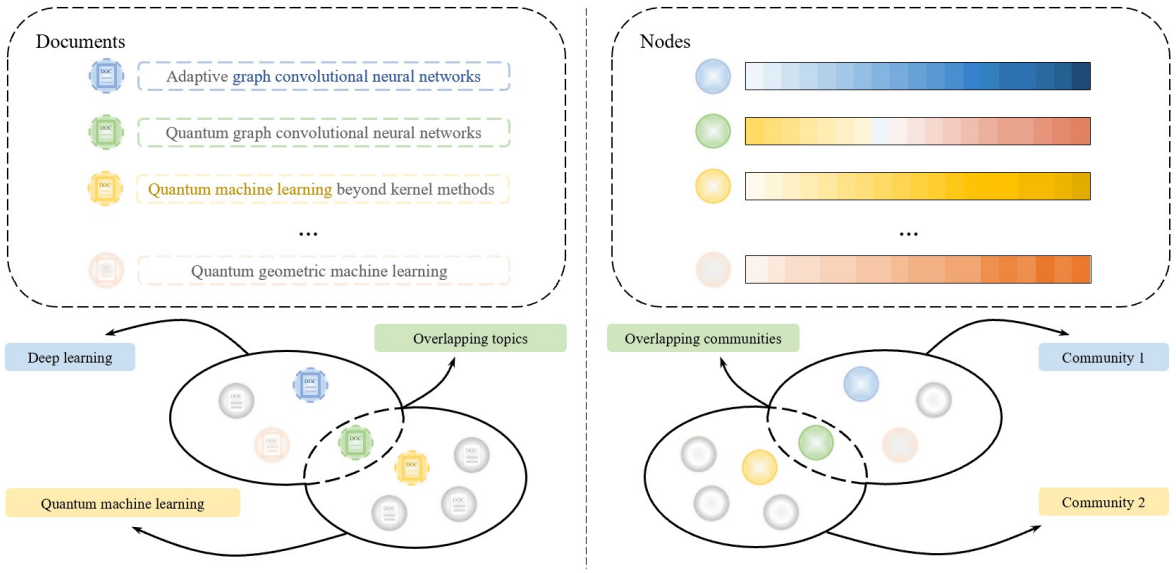

Fig. 2. Comparison of network nodes and their features with the thematic representation of documents and words.

decision accuracy of GNNs under extreme conditions. In the overlapping community detection phase, inspired by the Latent Dirichlet Allocation (LDA) method, commonly used in natural language processing to model "document–word" relationships [29], we analogize graph nodes to "documents" and their attribute features to "words." This analogy enables us to learn each node's distribution over a latent "semantic topic" space, from which we infer its community affinities and construct a candidate set of nodes for potential attacks. Additionally, based on the structural properties of the graph, we design three strategies to generate connections between fake nodes and real nodes. These strategies fully leverage the feature differences among the candidate nodes to enhance the overall perturbation effect. Finally, grounded in the message-passing mechanism of GNNs and incorporating a structural homophily constraint, we develop an effective feature generation approach for fake nodes that is applicable across various scenarios.

In summary, SNIA serves as a versatile data perturbation scheme that generates adversarial data prior to GNN training. As such, it is independent of specific GNN architectures, making it highly transferable across diverse models. The framework consists of two main stages: the data perturbation and the adversarial evaluation. During the data perturbation stage, we create fake nodes with disruptive features designed to interfere with the GNN training process. In the adversarial evaluation stage, we evaluate the effects of the perturbed data on the training of arbitrary GNN models. The implementation details for each module will be discussed in the following subsections.

B. Community Affinity Distribution

Earlier studies utilized gradient information from surrogate models or meta-learning frameworks to determine fake node connections. However, these techniques generate only a single perturbed edge per iteration, which results in considerable time and space complexities [10], [18], rendering them impractical for real-world applications. To overcome these limitations, we abandon surrogate model-based perturbation generation and instead derive perturbation information directly using the structural properties of the network.

To maximize the impact of perturbed data on the decision accuracy of GNNs, we expand our data perturbation strategy to involve connections with multiple nodes that exhibit multi-community preferences. As illustrated in Fig. 2, the structural relationships in real-world networks are often intricate—an entity with diverse preferences can simultaneously belong to multiple communities. Drawing inspiration from the "document–word" analogy, we integrate the LDA model into the graph structure to uncover the overlapping community structures among nodes. Within this modeling framework, the likelihood function for the generative process of a specific node i is defined as follows:

$$\mathcal{L} = \prod_{i=1}^{n} \int p\left(\theta_i|\alpha\right) \prod_{j=1}^{d} \sum_{k_{ij}} p\left(k_{ij}|\theta_i\right) p\left(x_{ij}|k_{ij},\beta\right) d\theta_i, \quad (6)$$

where $p\left(\theta_i|\alpha\right)$ signifies the community distribution for node i, following a Dirichlet distribution with α representing its hyperparameter; $p\left(k_{ij}|\theta_i\right)$ indicates the community distribution for the j_{th} feature of node i; k_{ij} represents the community corresponding to the j_{th} feature of node i; $p\left(x_{ij}|k_{ij};\beta\right)$ refers to the probability of feature x_{ij} being generated by community k_{ij}, where x_{ij} is the j_{th} feature of node i, and β denotes the parameter for the feature distribution of each community.

To approximate the intractable posterior $p(\theta, k \mid X)$, we employ variational inference with the following variational distribution:

$$q(\theta_i, k_i) = q(\theta_i|\gamma_i) \prod_{j=1}^{d} q(k_{ij}|\phi_{ij}), \quad (7)$$

Algorithm 1: Execution of the SNIA strategy

Input: The original attribute graph $G = (V, E, X, Y)$, community distribution parameter k, attack budget $\triangle$, candidate set T, overlapping community set C, and community affinity parameter ζ
Output: Poisoned graph $G' = (V', E', X')$

Initialize the undirected network G
Select the largest connected component in the original network
$\triangle E \leftarrow$ Compute the average degree of the network nodes
$\triangle F \leftarrow$ Determine the average value of non-zero features
/* Calculate the community distribution for all nodes n in the original graph. */
for *each* $i \in [0, n-1]$ **do**
    $\mathcal{L} \leftarrow$ Calculate the likelihood function using Eq. (7)
    $\mathcal{L} \leftarrow$ Optimize the calculation through variational distribution
    $\theta_{ik} \leftarrow$ Calculate the k-community distribution of nodes using Eq. (9)
end for
/* Initialize the overlapping communities and candidate node sets. */
$C \leftarrow 0$
$T \leftarrow 0$
if $\theta_{ik} > \zeta$ **then**
    $C \leftarrow$ Identify overlapping communities using Eq. (10)
    $T \leftarrow$ Assign nodes from the overlapping communities to the candidate set
end if
$V' \leftarrow$ Insert fake nodes into the original node set
$E' \leftarrow$ Generate perturbed edges using the attack method
/* Initialize the homophily parameters. */
$H \leftarrow 0$
while $i <$ *maximal iterations* **do**
    $X' \leftarrow$ Generate adversarial features using Eq. (14)
    $h_i \leftarrow$ Compute node similarity using Eq. (15)
    if $H < h_i$ **then**
        $X' \leftarrow$ Update
    end if
    $i \leftarrow i + 1$
end while
Update V', E', and X'
return $G' = (V', E', X')$

where γ_i stands for the community distribution parameter for node i, and ϕ_{ij} refers to the probability that the j_{th} feature is assigned to the community. This process achieves optimal parameter learning by minimizing the variational free energy function:

$$\mathcal{T} = \mathbb{E}_q[\log p(X, \theta, k|\alpha, \beta)] - \mathbb{E}_q[\log q(\theta, z)]. \tag{8}$$

After convergence, the community distribution for node i is derived by normalizing γ_i as follows:

$$\theta_i = \frac{\exp(\gamma_i)}{\sum_{k=1}^{K} \exp(\gamma_{ik})}. \tag{9}$$

Based on the community distribution θ_i of each node, we construct its overlapping community partition. Specifically, when the probability of a node belonging to a certain community exceeds the community affinity parameter ζ, the node is considered to fall under that community:

$$\mathcal{C} = \begin{cases} 1 & \text{if } \theta_i \geq \zeta, i \in \{1, 2, \ldots, k\} \\ 0 & otherwise \end{cases}. \tag{10}$$

The resulting candidate set $\mathcal{C}$ functions as a structure-aware prior for the subsequent design of perturbation strategies, providing a foundation for generating high-performance perturbations.

TABLE II
KEY CHARACTERISTICS OF FIVE REAL DATASETS

Datasets	Nodes	Edges	Features	Densities
Citeseer	2110	3668	3703	0.00165
Cora	2485	5069	1433	0.00164
Cora-ML	2810	7981	2879	0.00202
BlogCatalog	5196	171743	8189	0.12724
Flickr	89250	449878	500	0.00011

C. Structure-Aware Link Generation

We analyzed various connection strategies for perturbed data to assess how nodes of varying importance levels within the candidate set influence GNN performance. Based on node importance, the feature connections for fake nodes in SNIA were divided into three strategies: random connection (SNIA-R), maximum degree connection (SNIA-D), and maximum betweenness connection (SNIA-B). Each attack strategy selects nodes with different levels of importance from the candidate set to examine how different perturbation generation methods affect GNN classification outcomes.

In SNIA-R, the model randomly selects $\triangle E$ nodes from the candidate set to connect with the fake nodes. This connection strategy evaluates how nodes of varying levels of importance within the candidate set affect the decision-making process of the classifier, thereby assessing the robustness of the attack. In SNIA-D, the model prioritizes choosing $\triangle E$ nodes with the highest degree from the candidate set to establish fake node connections. The node degree is calculated as follows:

$$D_i = \sum_{j=1}^{n} \mathbb{1}(A_{ij}), \tag{11}$$

where $\mathbb{1}$ indicates the connection between the ith node and its neighboring nodes in the adjacency matrix A. This strategy assesses the impact of the most locally significant nodes within the candidate set on the classifier's performance. In SNIA-B, the model selects $\triangle E$ nodes with the highest betweenness centrality from the candidate set to establish fake node connections. The betweenness centrality of a node is determined as follows:

$$BC_i = \sum_{s \neq i \neq t} \frac{\sigma_{st}(i)}{\sigma_{st}}, \tag{12}$$

where σ_{st} signifies the number of shortest paths between nodes s and t, and $\sigma_{st}(i)$ implies the number of those paths passing through node i. This connection strategy evaluates the impact of globally important nodes within the candidate set on the classifier's performance.

D. Imperceptible Feature Generation

SNIA seeks to introduce subtle data perturbations that remain undetected by GNNs during training. Consequently,

TABLE III
BASELINE MODEL AND ITS RESPECTIVE ADJUSTMENTS IN EXPERIMENTAL VALIDATION

No.	Attack	Year	Target	Modification	Remark
1	Nettack	2018	Graph modification attack	Global injection attack	Disturbance sampling
2	AFGSM	2020	Target injection attack	Global injection attack	Global extension
3	G-NIA	2021	Target injection attack	Global injection attack	Global extension
4	GANI	2024	Global injection attack	—	No modification
5	LPGIA	2024	Global injection attack	—	No modification
6	SNIA-R	2024	Global injection attack	—	Our strategy
7	SNIA-D	2024	Global injection attack	—	Our strategy
8	SNIA-B	2024	Global injection attack	—	Our strategy

the features of injected fake nodes must closely match those of existing nodes. To achieve consistency across various node feature types, SNIA normalizes these features, with the normalization process formulated as follows:

$$X = \frac{X - X_{min}}{X_{max} - X_{min}}, \tag{13}$$

where X_{max} and X_{min} are the maximum and minimum values in the feature matrix X, respectively. As expressed in Eq. (2), the GNN aggregation scheme clearly operates within the two-hop neighborhoods. Consequently, SNIA restricts the feature similarity to the two-hop neighbors of the target node, ensuring consistency between fake and original nodes in the network. The feature generation process for fake nodes is outlined as follows:

$$X^{'}[\Delta F_i] = \frac{\sum_{j=1}^{\mathcal{N}} X[v, \Delta F_i]}{\sum_{j=1}^{\mathcal{N}} \mathbb{1}(X[v, \Delta F_i] \neq 0)}, \tag{14}$$

where ΔF represents the feature budget for the node, and $\mathcal{N}$ indicates the set of neighboring nodes of the target node.

According to Eq. (14), the SNIA method generates fake node features by allocating the feature attack budget ΔF. It iteratively selects feature values from the corresponding positions in neighboring nodes until the entire budget is exhausted. The resulting attributes of the fake nodes are derived as the mean values of the selected positions. For discrete features, the average feature value is consistently 1, whereas for continuous features, the value remains continuous after averaging. This method allows the SNIA scheme to effectively generate features that accommodate both discrete and continuous types, catering to a wide array of real-world applications.

Earlier studies have demonstrated that the adaptability of node injection attack techniques can, to a certain degree, disrupt the structural characteristics of the network [23]. In this study, we introduce a limitation on the homogeneity between the generated fake nodes and their neighboring nodes, thereby producing perturbations that are harder to identify. This limitation mandates that the fake nodes exhibit significant homogeneity with their neighbors. The approach for computing the homogeneity constraint is as follows:

$$h_i = cos\left(r_i, X_i'\right), \tag{15}$$

where cos denotes the cosine similarity, X_i' indicates the generated feature of fake node, and r_i is defined as follows:

$$r_i = \sum_{j=1}^{\mathcal{N}} \frac{1}{\sqrt{d_i}\sqrt{d_j}} X_j, \tag{16}$$

where d_i and d_j are the degree of the node i and j, respectively. This generation constraint enhances the feature generation process, ensuring local coherence between the fake node features and the original network feature.

E. Time Complexity Analysis

This section presents the pseudocode for the SNIA scheme in Algorithm 1 and examines its time complexity to highlight its effectiveness. The SNIA method comprises two key components: node connection perturbation and fake node feature generation. Specifically, the computational cost for generating perturbation edges is $O(ndk)$, where n indicates the number of network nodes, d denotes the feature dimension, and k signifies the number of planned community divisions. The computational cost for feature generation is $O(nd)$, where n implies the number of neighboring nodes. Consequently, the overall computational complexity of the SNIA algorithm exhibits $O(ndk)$.

Unlike other attack methods that involve training a surrogate model to produce perturbation data before attacking the target model, SNIA bypasses this traditional method, employing a more efficient perturbation generation method. Resultantly, the time complexity of our approach is solely determined by the data generation process, eliminating the need for surrogate model training and ensuring efficient attack execution.

IV. EXPERIMENTS

This section highlights two key components: the experimental setup and experimental analysis. Through comprehensive experimentation, we validate the effectiveness of the proposed approach. The experimental setup includes details on the five datasets, the baseline models, and the target models subjected to attack. The experimental analysis evaluates the performance of the SNIA scheme through comparative experiments, sensitivity analysis of parameters, and ablation studies.

TABLE IV
NODE CLASSIFICATION RESULTS OF THREE STANDARD GNN MODELS ACROSS FIVE DATASETS

Datasets	Victim	Clean	Nettack	AFGSM	GNIA	GANI	LPGIA	SNIA-R	SNIA-D	SNIA-B
Citeseer	GCN	0.7447	0.7198	0.7476	0.7286	0.7156	0.7306	**0.7049**	0.7110	0.7085
	GAT	0.7417	0.7314	0.7402	0.7393	0.7328	0.7379	0.7037	0.7014	**0.6955**
	SGC	0.7441	0.7320	0.7494	**0.7204**	0.7470	0.7350	0.7287	0.7251	0.7268
Cora	GCN	0.8490	0.8305	0.8375	0.8053	0.8410	0.8387	**0.8003**	0.8014	0.8048
	GAT	0.8319	0.8395	0.8342	0.8265	0.8224	0.8378	0.7927	**0.7801**	0.7947
	SGC	0.8314	0.8290	0.8314	0.8133	0.8134	0.8282	0.8129	**0.8089**	0.8133
Cora-ML	GCN	0.8581	0.8479	0.8456	0.8425	0.8431	0.8541	0.8274	**0.8105**	0.8193
	GAT	0.8536	0.8421	0.8350	0.8190	0.8296	0.8440	0.8078	**0.7945**	0.8033
	SGC	0.8372	0.8280	0.8324	0.8243	0.8211	0.8367	**0.8185**	0.8211	0.8207
BlogCatalog	GCN	0.7670	0.7589	0.7654	0.7738	0.7479	0.7596	0.7168	0.7224	**0.7139**
	GAT	0.6579	0.6233	0.6263	0.6310	0.6259	0.6365	0.5711	**0.5708**	0.5823
	SGC	0.7426	0.7410	0.7420	0.7410	0.7412	0.7411	0.7383	0.7380	**0.7375**
Flickr	GCN	0.4740	0.4749	0.4735	0.4452	0.4719	0.4653	0.4279	0.4275	**0.4268**
	GAT	0.4710	0.4698	0.4644	0.4551	0.4692	0.4661	0.4226	**0.4217**	0.4226
	SGC	0.4342	0.4226	0.4273	0.4230	0.4236	0.4226	**0.4197**	0.4211	0.4208

A. Experimental Setup

1) Datasets: We considered five representative real-world network datasets for our experiments: Citeseer, Cora, Cora-ML, BlogCatalog, and Flickr. These datasets were selected to cover various network scales, data types, and densities, encompassing diverse practical application scenarios. To mitigate the influence of independent communities or isolated nodes on the experimental results, we followed the experimental setup from prior studies, concentrating on the largest connected component of the network. The relevant metrics for these specific datasets are provided in Table II.

2) Victim Models: To evaluate the computational performance of the proposed model on various victim GNNs, we selected two prominent sets of target models widely recognized in the field of adversarial attacks. The first group consists of key GNNs, namely GCN [26], graph attention network (GAT) [30], and simplified graph convolution (SGC) [31]. The second group comprises defensive GNNs, specifically the node similarity preserving model (SimpGCN) [32] and the perturbed edge detection model (GNNGuard) [33], developed to counteract the effects of adversarial data perturbations on GNNs prediction accuracy. These two GNN groups were chosen to represent practical scenarios encountered by attack strategies.

3) Baseline Models: To validate the effectiveness of the proposed method, we compared the SNIA model with five well-established attack methods, divided into three categories. The first category includes graph modification attacks, exemplified by network attack (Nettack) [12]. The second category consists of target node attacks, represented by adversarial fast gradient sign method (AFGSM) [19] and generalizable node injection attack (G-NIA) [20]. The final category includes global node attacks, represented by global attack node injection (GANI) [21] and label-propagation-based global injection attack (LPGIA) [34]. The selected baseline model comprehensively incorporates recent graph adversarial attack strategies to examine and compare the variations among these methods. Since various strategies target distinct objectives, we adjusted some modifications to suit global injection attacks. These modifications are outlined in Table III.

4) Parameter Settings: To ensure consistency in our experimental results, we adhered to the settings established in prior studies regarding dataset split ratios and edge attack budgets [34]. Specifically, the dataset was partitioned into 10% for training, 10% for validation, and 80% for testing. The edge attack budget is determined based on the network density: if the density is below 0.1, we set the edge budget to the average node degree, as per [34]. To minimize attention from the GNN model in dense networks (i.e., when the density exceeds 0.1), the edge budget is set to the minimum node degree within the candidate node set. Moreover, since altering all feature dimensions of a node is impractical in real-world scenarios, we adopted the approach in [21] for the node feature attack budget, which is defined as the average number of non-zero features per node in the network. The number of communities k is set equal to the number of node label classes, and the community affinity parameter ζ is fixed at 0.4. In our validation experiments, the node attack budget for all algorithms is uniformly set to 1, while other parameters follow the default configurations of each respective method.

B. SNIA Performance Under Regular GNNs

This section evaluates the performance of all baseline models across three representative regular GNN architectures using five datasets. The detailed experimental results are shown in Table IV. These findings demonstrate that SNIA outperforms all other attack strategies, except for the SGC model on the Citeseer dataset, where it slightly lags by 0.47% behind G-NIA. This minor difference is due to the advantages of the white-box attack executed by G-NIA.

All three SNIA-based attack strategies exhibit significant perturbation effects, with a minimal gap between them in an overall perturbation of up to 8.71%. Although the optimal

TABLE V
NODE CLASSIFICATION RESULTS OF TWO DEFENSIVE GNN MODELS ACROSS FIVE DATASETS

	Citeseer		Cora		Cora-ml		BlogCatalog		Flickr	
	SimpGCN	GNNGuard	SimpGCN	GNNGuard	SimpGCN	GNNGuard	SimpGCN	GNNGuard	SimpGCN	GNNGuard
Clean	0.7571	0.7292	0.8244	0.8189	0.8576	0.8563	0.9165	0.7643	0.4671	0.4774
Nettack	0.7351	0.7233	0.8284	0.8189	0.8381	0.8416	0.8703	0.7333	0.4429	0.4696
AFGSM	0.7476	0.7287	0.8199	0.8355	0.8501	0.8389	0.8703	0.7604	0.4369	0.4648
G-NIA	0.7393	0.7233	0.8269	0.8104	0.8505	0.8629	0.9014	0.7537	0.4596	0.4656
GANI	0.7322	0.7310	0.8013	0.8085	0.8416	0.8438	0.8891	0.7623	0.4342	0.4687
LPGIA	0.7452	0.7254	0.8283	0.8349	0.8509	0.8521	0.9003	0.7575	0.4601	0.4703
SNIA-R	0.7150	0.7085	0.7927	**0.7928**	0.8194	**0.8133**	**0.8433**	0.7241	**0.4221**	**04225**
SNIA-D	**0.7043**	0.7126	**0.7902**	0.8003	**0.8185**	0.8233	0.8573	**0.7171**	0.4240	0.4297
SNIA-B	0.7174	**0.7079**	0.7932	0.8048	0.8278	0.8238	0.8545	0.7244	0.4226	0.4252

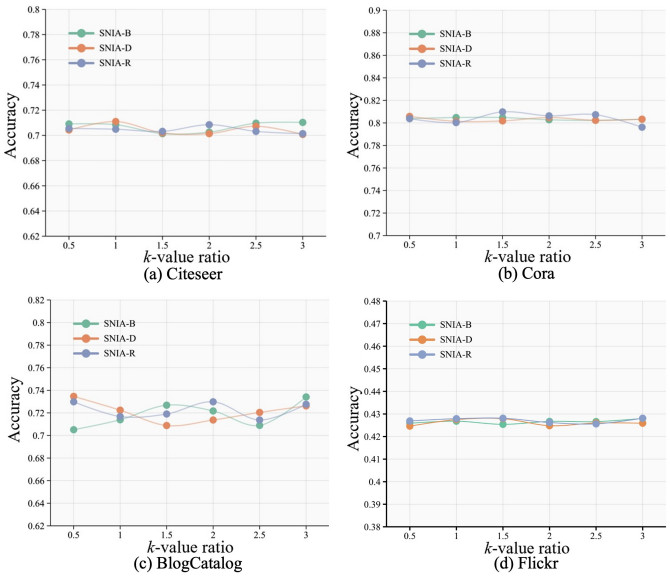

Fig. 3. Analysis of parameter sensitivity in the SNIA approach.

perturbation effect varies across different strategies, each one outperforms the latest attack method. Furthermore, SNIA strategies consistently produce stable perturbations across all models, despite differences in how GNNs process data. Experimental results on regular GNNs validate the robust generalization capability of the SNIA scheme.

C. SNIA Performance Under Defensive GNNs

To demonstrate the robustness of SNIA and the generalization capability of its attack strategies, we evaluated two prominent defense-oriented GNNs. The detailed experimental results are presented in Table V. In experiments with defensive GNNs, SNIA along with its attack methods displayed consistent attack performance, indicating that, under identical data and model conditions, the effectiveness of the attacks remains highly stable across different attack methods.

Similarly to the results observed on regular GNNs, the optimal perturbation effects are observed across SNIA-R, SNIA-D, and SNIA-B, respectively, with each attack strategy performing exceptionally well on various datasets. Even in datasets and defensive models where other attack strategies falter, SNIA achieves a perturbation effect of 7.32%, reinforcing its robust generalization capability. Coupled with the experimental results outlined in Table IV, the proposed SNIA scheme demonstrates strong attack effectiveness against both regular and defensive GNNs, highlighting its ability to bypass defense mechanisms.

D. Ablation Studies

This section presents ablation studies to evaluate the contribution of different mechanisms within SNIA. The SNIA model consists of two key modules: structure-aware node selection for edge generation and imperceptible neighbor similarity perturbation for feature generation. We remove one or both mechanisms to analyze the impact of each module on the attack's performance. The SNIA_{l-f} variant removes all modules, generating perturbation edges and features randomly. SNIA_{l} omits only the edge generation module, selecting edge connections randomly from the entire network. In contrast, SNIA_{f} eliminates the feature generation module, using a random feature generation approach.

The experimental outcomes of the ablation study are presented in Table VI. In the SNIA attack scheme, the feature generation module offers a more subtle and effective attack than the edge generation module; however, the absence of global perturbation reduces the visibility of the attack. The edge generation strategy ensures a reliable attack effect. Nevertheless, without constraining the features of fake nodes, the perturbed features appear too abrupt. As a result, neither module achieves a powerful attack effect. In contrast, the combined use of both attack modules demonstrates a robust attack across all datasets.

E. Parameter Sensitivity Analysis

To assess the effect of community count on the SNIA attack, we performed a parameter sensitivity analysis, examining how different values of the community parameter k impact the actual effects of the attack. In this section, we used the base community number as the starting point, varying other community values between 0.5 and 3 in increments of 0.5. Since the Cora and Cora-ML datasets are derived from different processing versions of the same raw dataset and yield

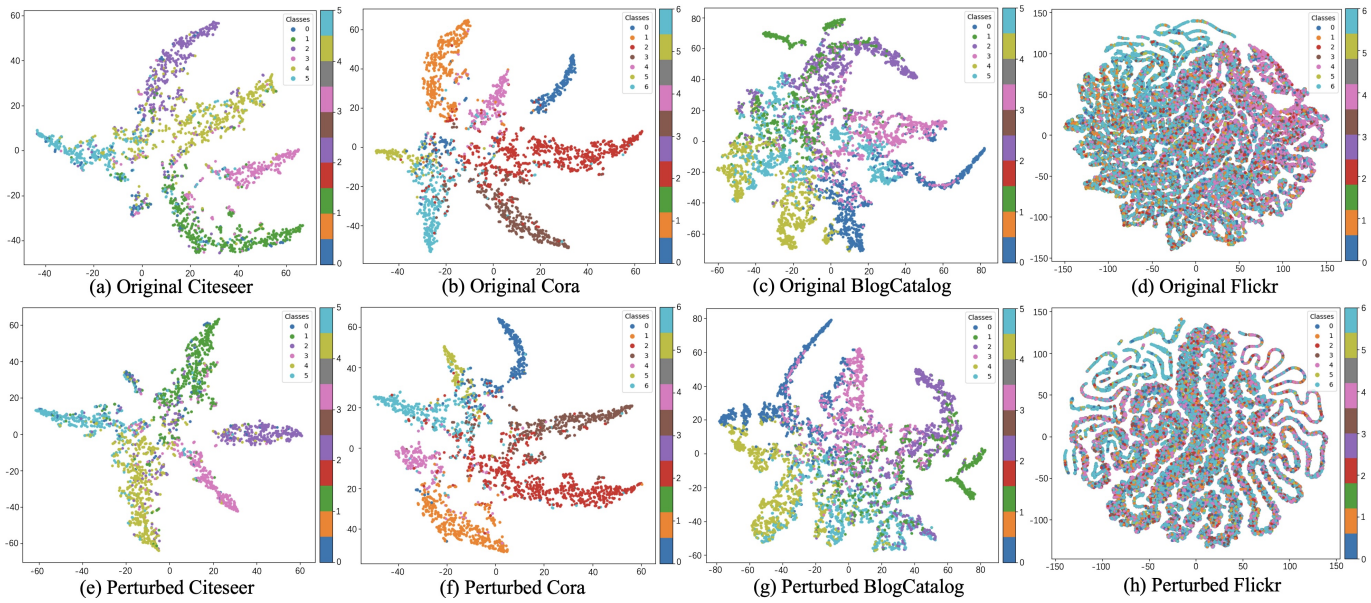

Fig. 4. Visualization results of the GNN-based node classification predictions before and after the implementation of the SNIA attack strategy.

TABLE VI
OUTCOMES OF ABLATION EXPERIMENTS USING THE SNIA APPROACH

Modules	Citeseer	Cora	Cora-ML	BlogCatalog	Flickr
Clean	0.7571	0.8244	0.8576	0.9165	0.4671
SNIA$_{l-f}$	0.7440	0.8184	0.8514	0.8992	0.4597
SNIA$_{l}$	0.7268	0.7997	0.8425	0.8794	0.4525
SNIA$_{f}$	0.7352	0.7952	0.8394	0.8621	0.4502
SNIA	**0.7043**	**0.7902**	**0.8185**	**0.8573**	**0.4240**

consistent experimental results, we present only the results for the Citeseer, Cora, BlogCatalog, and Flickr datasets in Fig. 3. These results illustrate the performance of SNIA under diverse parameter settings.

The experimental findings indicate that all three attack strategies of SNIA consistently maintain their effectiveness across all datasets. The perturbation results remain stable and are unaffected by variations in the number of communities k. This further confirms the robustness of the SNIA method, which continues to deliver effective attacks despite changes in network structure or community assignments. Furthermore, this highlights the adaptability of the SNIA approach, stressing its relevance in real-world scenarios.

F. Visualization Studies

In this section, we utilize the t-SNE algorithm [35] to visualize data representations post-model training, aiming to assess the impact of SNIA on GNN classification performance before and after the attack. The datasets selected align with those used in the parameter sensitivity experiments. Figure 4 presents the t-SNE visualization results of SNIA across four datasets: Figs. 4(a)–(d) depict the training outcomes of clean GNN models on the Citeseer, Cora, BlogCatalog, and Flickr datasets, while Figs. 4(e)–(h) display the corresponding results after applying SNIA perturbations on the same datasets.

The observations from the experiments reveal that on the Citeseer and Cora datasets, GNNs effectively classify nodes in clean graphs, showing clear class boundaries and compact intra-class clustering. However, following the SNIA attack, these class boundaries become significantly blurred, and intra-class clusters are noticeably dispersed. For the BlogCatalog and Flickr datasets, while the initial class boundaries are less distinct, nodes within each class still exhibit some degree of cohesion. After the SNIA attack, classification confusion increases markedly, with a pronounced effect observed in the Flickr dataset. These visualization results illustrate that SNIA severely degrades classification performance across various datasets, demonstrating that even the injection of a single node can compromise the accurate classification of a substantial number of nodes.

G. Graph Distribution Analysis

As follows, we analyze the extent to which different attack strategies perturb the graph structure. Our evaluation is conducted from two perspectives. First, we visualize the homophily distributions of the original and poisoned graphs to intuitively illustrate the differences in perturbation. Second, we quantitatively measure structural changes by computing the KL divergence [36] between the original and poisoned graphs. To better observe the impact, we conducted experiments on the Citeseer dataset with all parameters kept consistent. Figure 5 showcases a thorough graphical depiction of our analysis.

Previous evaluations suggest that stronger manipulations of graph structures can significantly impair the classification performance of GNNs. However, such manipulations are also more susceptible to detection and mitigation by GNN-based defense mechanisms. The distribution results reveal that, irrespective of the specific SNIA variant employed, SNIA introduces the least structural perturbation compared to other state-of-the-art attack methods. This implies that by incorporating structural constraints, SNIA maintains high attack effectiveness while minimizing disruption to the original

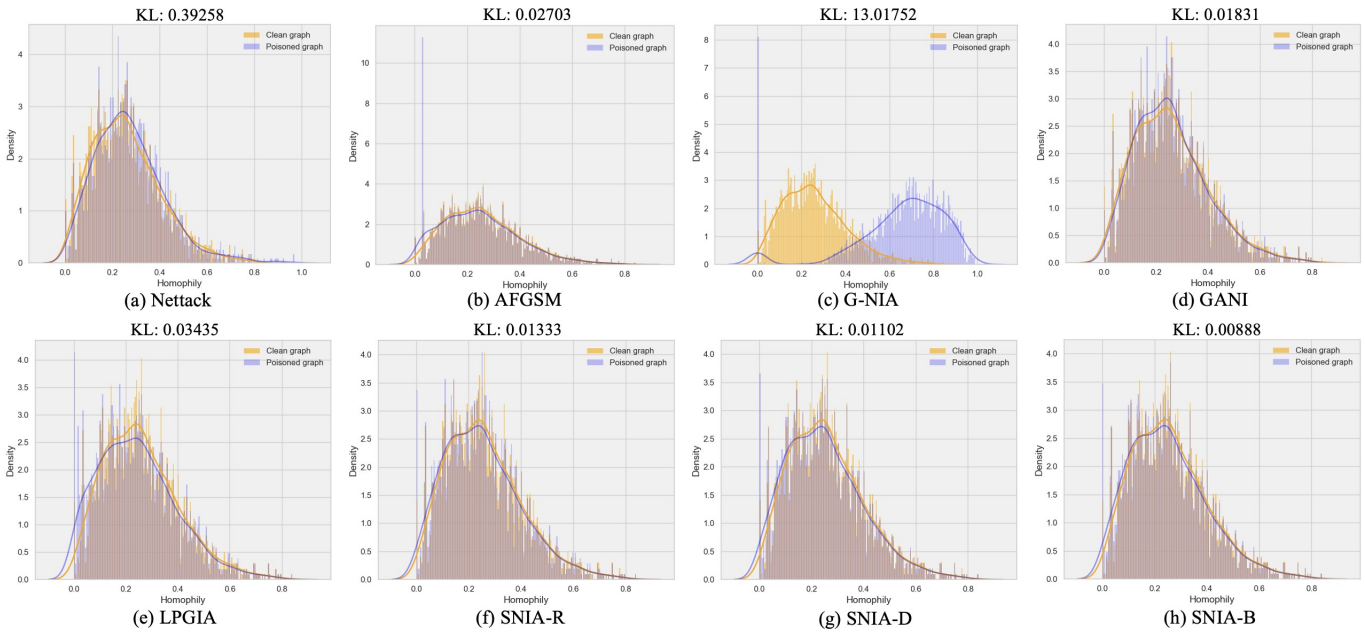

Fig. 5. Homophily distribution under different attack strategies compared to the distribution without attack.

graph structure. These findings further manifest that SNIA's efficacy in disrupting GNNs does not depend on damaging the graph structure; instead, preserving that structure is crucial for evading detection by GNN defenses.

V. Conclusion

This study explores the impact of node injection techniques on the training accuracy of GNNs, with due consideration of network structure. In contrast to prior research, our proposed algorithm achieves a significant reduction in GNN predictive accuracy even under minimal attack conditions. We develop the attack strategy by leveraging network topology and targeting nodes prone to belonging to overlapping communities, thereby maximizing the effectiveness of the injected node. To evade detection by the GNN model, we impose homogeneity constraints during the feature generation process. This ensures that the injected fake node closely resembles its neighbors, enhancing the stealthiness of the attack. Extensive experimental results verify that this method effectively disrupts GNN models, including those equipped with defense mechanisms.

Numerous studies on graph injection attacks have shown that perturbing graph-structured data can considerably compromise the robustness of GNN models. However, a prevalent challenge with node injection attacks is the mismatch between the injected perturbation nodes and the real nodes in the original graph. Such discrepancies may result in high node reconstruction errors, creating potential avenues for detecting and defending against adversarial perturbations. Future research should thus focus on minimizing node reconstruction errors and strengthening the consistency between node features and graph structure, so as to enhance the robustness of GNNs in adversarial settings.

References

[1] S. Ryoma, Y. Makoto, and K. Hisashi, "Constant time graph neural networks," *ACM Transactions on Knowledge Discovery from Data*, vol. 16, no. 5, pp. 1–31, 2022.

[2] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, 2005, pp. 729–734.

[3] Y. Liu, S. Zhao, X. Wang, L. Geng, Z. Xiao, and J. C.-W. Lin, "Self-consistent graph neural networks for semi-supervised node classification," *IEEE Transactions on Big Data*, vol. 9, no. 4, pp. 1186–1197, 2023.

[4] S. Vashishth, S. Sanyal, B. Ray, and P. Talukdar, "Composition-based multi-relational graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, 2020.

[5] J. Wang, P. Chen, B. Ma, J. Zhou, Z. Ruan, G. Chen, and Q. Xuan, "Sampling subgraph network with application to graph classification," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 4, pp. 3478–3490, 2021.

[6] L. Sang, M. Xu, S. Qian, and X. Wu, "Adversarial heterogeneous graph neural network for robust recommendation," *IEEE Transactions on Computational Social Systems*, vol. 10, no. 5, pp. 2660–2671, 2023.

[7] B. Pietro, P. Niccolò, B. Asma, S. Franco, M. Marco, and B. Monica, "Composite graph neural networks for molecular property prediction," *International Journal of Molecular Sciences*, vol. 25, no. 12, p. 6583, 2024.

[8] Y. Zhang, N. Yuan, Z. Zhang, J. Du, T. Wang, B. Liu, A. Yang, K. Lv, G. Ma, and B. Lei, "Unsupervised domain selective graph convolutional network forpreoperative prediction of lymph node metastasis in gastric cancer," *Medical Image Analysis*, vol. 79, p. 102467, 2022.

[9] T. T. Nguyen, D. K. Q. Nguyen, T. N. Thanh, T. H. Thanh, H. V. Viet, L. N. Phi, J. Jun, and V. H. N. Quoc, "Poisoning GNN-based recommender systems with generative surrogate-based attacks," *ACM Transactions on Information Systems*, vol. 41, no. 3, pp. 1–24, 2023.

[10] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *Proceedings of the International Conference on Learning Representations*, 2018.

[11] L. Sun, Y. Dou, C. Yang, K. Zhang, J. Wang, P. S. Yu, L. He, and B. Li, "Adversarial attack and defense on graph data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 8, pp. 7693–7711, 2022.

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

[13] B. Wang and N. Z. Gong, "Attacking graph-based classification via manipulating the graph structure," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2023–2040.
[14] X. Lin, C. Zhou, H. Yang, J. Wu, H. Wang, Y. Cao, and B. Wang, "Exploratory adversarial attacks on graph neural networks," in *Proceedings of the IEEE International Conference on Data Mining*, 2020, pp. 1136–1141.
[15] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *Proceedings of the International Conference on Machine Learning*, 2018, pp. 1115–1124.
[16] M. Zhang, L. Hu, C. Shi, and X. Wang, "Adversarial label-flipping attack and defense for graph neural networks," in *Proceedings of the IEEE International Conference on Data Mining*, 2020, pp. 791–800.
[17] Y. Sun, S. Wang, X. Tang, T. Hsieh, and V. G. Honavar, "Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach," in *Proceedings of the Web Conference*, 2020, pp. 673–683.
[18] Y. Chen, Z. Ye, Z. Wang, and H. Zhao, "Imperceptible graph injection attack on graph neural networks," *Complex and Intelligent Systems*, vol. 10, no. 1, pp. 869–883, 2023.
[19] 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.
[20] 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 and Knowledge Management*, 2021, pp. 1794–1803.
[21] J. Fang, H. Wen, J. Wu, Q. Xuan, Z. Zheng, and C. Tse, "GANI: global attacks on graph neural networks via imperceptible node injections," *IEEE Transactions on Computational Social Systems*, vol. 11, no. 4, pp. 5374–5387, 2024.
[22] 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 ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021, pp. 2461–2471.
[23] Y. Chen, H. Yang, Y. Zhang, K. Ma, T. Liu, B. Han, and J. Cheng, "Understanding and improving graph injection attack by promoting unnoticeability," in *Proceedings of the International Conference on Learning Representations*, 2022.
[24] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 23, pp. 828–841, 2019.
[25] 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.
[26] T. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, 2017.
[27] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.
[28] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," in *Proceedings of the IEEE International Conference on Data Mining*, 2013, pp. 1151–1156.
[29] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.
[30] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proceedings of the International Conference on Learning Representations*, 2018.
[31] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the International Conference on Machine Learning*, 2019, pp. 6861–6871.
[32] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, "Node similarity preserving graph convolutional networks," in *Proceedings of the ACM International Conference on Web Search and Data Mining*, 2021, pp. 148–156.
[33] X. Zhang and M. Zitnik, "GNNGuard: Defending graph neural networks against adversarial attacks," *Neural Information Processing Systems*, vol. 33, pp. 9263–9275, 2020.
[34] 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.
[35] L. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 2605, pp. 2579–2605, 2008.
[36] J. M. Joyce, "Kullback-leibler divergence," *Springer Berlin Heidelberg*, 2011.

Fei Yan is a professor at the School of Computer Science and Technology, Changchun University of Science and Technology, China. He is now a visiting professor at the Department of Computer Science and Technology, Tsinghua University, China. He received his doctorate in engineering and worked as a postdoctoral fellow afterward in the Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology, Japan. He is serving as an Associate Editor of the Information Sciences journal and Journal of Advanced Computational Intelligence and Intelligent Informatics. He has published more than 100 papers in the fields of computational intelligence, quantum information processing, machine learning, and health informatics.

Yanlong Tang graduated from the School of Computer Science, Qinghai Normal University, China, with a Master's degree. He is currently a research assistant at the School of Computer Science and Technology, Changchun University of Science and Technology, China. His research interests include natural language processing, representation learning, and graph neural networks.

Witold Pedrycz (F'98) is a professor in the Department of Electrical and Computer Engineering, University of Alberta, Canada. He was elected a foreign member of the Polish Academy of Sciences and a Fellow of the Royal Society of Canada. He received a prestigious Norbert Wiener award from the IEEE Systems, Man, and Cybernetics Society. He is a recipient of the IEEE Canada Computer Engineering Medal, a Cajastur Prize for Soft Computing from the European Centre for Soft Computing, a Killam Prize, and a Fuzzy Pioneer Award from the IEEE Computational Intelligence Society. He is the Editor-in-Chief of Information Sciences (*Elsevier*), WIREs Data Mining and Knowledge Discovery (*Wiley*), and Journal of Data, Information and Management (*Springer*), etc. He has total citations of more than 110,000, an h-index of 139, and an i10-index of 1,472 based on Google Scholar.

Kaoru Hirota (LM'16) is an emeritus professor at the Tokyo Institute of Technology in Japan and a distinguished professor at the School of Automation, Beijing Institute of Technology, China. From 2015 to 2021, he served as the director of the Beijing representative office of the Japan Society for the Promotion of Science (JSPS). He is a fellow and former president of the International Fuzzy Systems Association (IFSA) and has also served as the president of the Japan Society for Fuzzy Theory and Systems (SOFT). He has received several prestigious awards, including the Banki Donat Medal, the Henri Coanda Medal, the Chinese Friendship Medal, and the Grigore MOISIL Award. He is currently the Editor-in-Chief of the Journal of Advanced Computational Intelligence and Intelligent Informatics, as well as an advisory editor and associate editor for several prestigious international journals. He has organized numerous international conferences and workshops as a Founding and General Chair.