UNGIA: Unnoticeable Injection Attack on Graph Neural Networks

Zixuan Wang[1], Ningyun Chen[1], Leo Yu Zhang[2], Donglong Chen[1]
[1]Guangdong Provincial Key Laboratory of IRADS,
[1]Beijing Normal-Hong Kong Baptist University, Zhuhai, China,
[2]Griffith University, Brisbane, Australia
r130026148@outlook.com, nychen013@outlook.com, leo.zhang@griffith.edu.au, donglongchen@bnbu.edu.cn

***Abstract*—Graph neural networks excel in classification and prediction tasks but are highly susceptible to adversarial attacks. Among attack strategies, graph injection attacks (GIA) are more practical than graph modification attacks as they avoid altering the original graph structure. However, existing GIA methods often sacrifice stealth for effectiveness, making fake nodes easily detectable and reducing their impact. To address this, we propose UNGIA, a steganographic GIA method. UNGIA employs clustering to identify anchor nodes and an adaptive generator to learn the feature distribution of original nodes, enabling the creation of indistinguishable fake nodes. To preserve graph homophily—the principle that similar nodes tend to connect—potential links are established using a mask learning mechanism. Additionally, a two-layer optimization framework is designed to balance stealth and attack effectiveness. Experiments on Cora, OGB-arxiv, and Reddit datasets demonstrate that UNGIA is covert, effective, and transferable, outperforming state-of-the-art methods in both effectiveness and adaptability. On average, UNGIA further reduces classification accuracy by about 3.3 percentage points compared with the best baseline methods.**

***Index Terms*—Graph neural networks, Graph injection attack, Unnoticeable perturbations**

I. Introduction

Graph Neural Network (GNN) is a deep learning model specially designed for graph-structured data, which is widely used in social networks, molecular structures, and knowledge graphs. However, GNN is vulnerable to adversarial attacks, such as modifying node properties or injecting fake nodes [2, 4]. Adversarial attacks are mainly divided into graph modification attacks (GMA) and graph injection attacks (GIA). In contrast, GIA does not need to access raw data and is more concealed, making it closer to real-life scenarios.

Prior GIA methods range from gradient-based (AFGSM [5]) and optimization-driven (NICKI [14], G-NIA [20]) to reinforcement learning-based approaches (NIPA [7], GA2C [8]). Recent works have focused on improving stealth by using feature smoothing (Zou et al. [21]) or aligning with normal node distributions (CANA [22]). However, many still struggle to balance stealth and effectiveness, especially on large graphs. While many GIAs prioritize attack success rates, they often overlook imperceptibility, leading to more detectable fake nodes. Chen et al. [23] highlighted that GIA's flexibility disrupts graph homogeneity, reducing similarity between neighboring nodes and increasing detection risks.

Crafting GIA strategies that are both stealthy and effective remains a significant challenge. GIAs continue to face the following issues: (1) **Fake node feature confusion:** Abnormal features of fake nodes are easy to detect; (2) **Homophily Disruption:** Injected nodes and their links often disrupt the graph's natural homophily (i.e., the property that nodes with similar features or the same labels connect to each other), making the perturbations structurally detectable and reducing the attack's stealth. Furthermore, with large data sets, the effectiveness of GIA methods drop significantly under budget constraints. In light of the aforementioned challenges, this article aims to design a highly concealable GIA method aimed at enhancing concealment while maintaining a high attack success rate.

The main contributions of this work are as follow:

- To enhance the efficiency and effectiveness of the attack, a clustering strategy was adopted to select representative nodes as key inputs for model learning.
- To achieve a balance between homogeneity and attack effectiveness while enhancing concealment, we propose a two-layer optimized node generator with homogeneity constraints. The generator is trained to learn the feature distribution based on the original node features.
- The node generator employs a bi-level optimization mechanism: the inner-layer optimization trains the agent GNN model on the perturbed graph to minimize prediction error, while the outer-layer optimization ensures homogeneity and attack effectiveness when injecting fake nodes and links, thereby reducing the risk of detection.
- The model demonstrates superior attack effectiveness on both classic GNNs and defense models compared to the baseline. It performs consistently well across multiple datasets and exhibits significant advantages, including strong transferability to large datasets, and on average

This work is supported in part by Scientific Research Innovation Capability Support Project for Young Faculty (SRICSPYF-BS2025137), Guangdong Provincial Key Laboratory of IRADS (2022B1212010006), Guangdong and Hong Kong Universities "1+1+1" Joint Research Collaboration Scheme (2025A0505000001), Guangdong Basic and Applied Basic Research Foundation (2024A1515011274), Guangdong Province General Universities Key Field Project (New Generation Information Technology) (2023ZDZX1033), and UIC Research Grant (UICR04202401-21). Ningyun Chen is also supported by the Konrad Zuse School of Excellence in Learning and Intelligent Systems (ELIZA, https://eliza.school) through the DAAD programme Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the Federal Ministry of Education and Research.

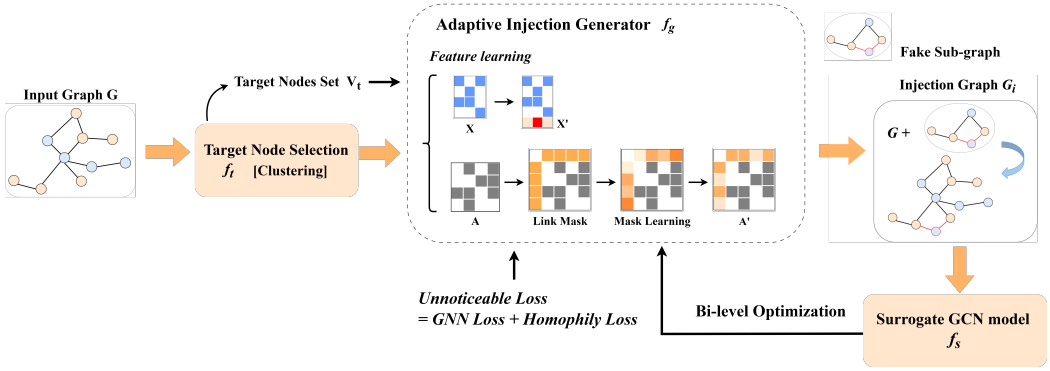


Fig. 1. Schematic diagram of UNGIA. Given an input graph G, target node selection is first performed by clustering. Then, the adaptive injection generator f_g generates a pseudo subgraph by performing link and mask learning through feature learning. This pseudo subgraph is injected into the original graph G to obtain the enhanced graph G^+. In the two-layer optimization process, the imperceptible loss is calculated by combining the GNN loss and the homogeneity loss. Finally, train the agent GCN model f_s based on the modified graph.

further reduces classification accuracy by about 3.3 percentage points compared with the best baseline methods.

II. Methodology

Figure 1 describes the specific construction process of UNGIA. Specifically, we use a clustering approach to identify representative nodes in the graph, which serve as anchors for the links of the fake nodes. An adaptive injection generator is trained to generate the initial features of the fake nodes and potential links between fake and original nodes. Through bi-level homogeneity constraint optimization, we balance homogeneity and attack effectiveness.

A. Anchor Node Selection via Clustering

To improve attack efficiency on large graphs, we avoid random anchor selection. Instead, we identify structurally representative nodes to serve as anchors for injected nodes. The core idea is to find nodes that are central to a local community but are not overly prominent, making them effective yet inconspicuous targets.

Let the graph be $G = (A, X)$ with adjacency matrix $A \in \{0,1\}^{|V|\times|V|}$ and node feature matrix $X \in \mathbb{R}^{|V|\times d}$. We first generate structure-aware embeddings using a GCN encoder, $Z = \text{GCN}(A, X)$, which captures both feature and topological information. These embeddings are then grouped into k clusters $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ with corresponding centroids $\{\mu_1, \ldots, \mu_k\}$ using the K-means algorithm. The number of clusters, k, is treated as a hyperparameter.

Within each cluster $\mathcal{C}_i$, we define a score to evaluate the suitability of each node v_j as an anchor. Our goal is to avoid nodes that are too central, as they are often core to the class structure and well-defended. To mitigate this, we introduce a degree-based penalty to our selection score. The score for node $v_j \in \mathcal{C}_i$ is defined as:

$$\text{Score}(v_j) = \frac{\|\mathbf{z}_j - \mu_i\|_2^2}{(d_j)^p} \tag{1}$$

where $\|\mathbf{z}_j - \mu_i\|_2^2$ measures the distance from the centroid, d_j is the degree of node v_j, and p is a hyperparameter controlling the penalty strength. This score favors nodes that are representative of their cluster (low distance) but not overly connected (low degree). For each cluster, we select the node with the minimum score as an anchor: $v_{\text{selected}} = \arg\min_{v_j \in \mathcal{C}_i} \text{Score}(v_j)$.

Algorithm 1 Cluster-Based Node Selection

Require: Graph $G = (A, X)$, budget b, clusters k, penalty p, weight λ.
Ensure: Selected nodes $\mathcal{S}$.

$Z \leftarrow \text{GCN}(A, X)$
$\{\mathcal{C}_1, ..., \mathcal{C}_k\}, \{\mu_1, ..., \mu_k\} \leftarrow \text{K-means}(Z, k)$
$\mathcal{S} \leftarrow \emptyset$
while $|\mathcal{S}| < b$ **do**
  for each cluster $\mathcal{C}_i$ **do**
    for each node $v_j \in \mathcal{C}_i$ **do**
      $\text{Rep}(v_j) \leftarrow \lambda \cdot \|\mathbf{z}_j - \mu_i\|_2^2$
      $d_j \leftarrow \text{degree}(v_j)$
      $\text{Score}(v_j) \leftarrow \text{Rep}(v_j)/(d_j)^p$
    end for
    $v_{\text{sel}} \leftarrow \arg\min_{v_j \in \mathcal{C}_i} \text{Score}(v_j)$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{v_{\text{sel}}\}$
  end for
end while
return $\mathcal{S}$

B. Adaptive Injection Generator

Given the selected anchor nodes, we design an adaptive generator f_g, implemented as a two-layer MLP, to create a budget of b fake nodes.

Feature Generation. The generator learns the underlying feature distribution from the anchor nodes. To ensure the generated features are inconspicuous, we enforce similarity to the real nodes' features via a feature similarity loss:

$$L_{\text{feature}} = \frac{1}{b}\sum_{i=1}^{b} \left\| x_i^{\text{fake}} - \bar{x}_{\text{anchor}} \right\|_2^2, \tag{2}$$

where x_i^{fake} is the feature of the i-th fake node and $\bar{x}_{\text{anchor}}$ is the mean feature vector of the selected anchor nodes.

Algorithm 2 Bi-Level Optimization Training for UNGIA

Require: Graph G, surrogate GNN f_s, generatorf_g, budget b.
Ensure: Optimized generator parameters θ_g.

```
Select anchor nodes using Algorithm 1.
for each training epoch do
    // Upper-Level: Generate Perturbation
    Generate fake node features and links using f_g(θ_g).
    Construct perturbed graph G' = (A', X').
    // Lower-Level: Update Surrogate Model
    Update θ_s by descending ∇_{θ_s} L_s(G', θ_s).
    // Upper-Level: Update Generator
    Compute L_total = L_attack + λL_h + γL_feature.
    Update θ_g by descending ∇_{θ_g} L_total.
end for
return θ_g.
```

Link Generation with Mask Learning. To form connections, the generator outputs a vector of logits for each fake node. A Gumbel-Softmax trick is then used to sample a discrete adjacency vector (a binary mask), effectively implementing a *mask learning mechanism*. It allows for differentiable selection of edges.

Overall Objective. The generator is trained to minimize a composite loss function that balances attack effectiveness and stealth:

$$L_{\text{total}} = L_{\text{attack}} + \lambda L_{\text{homophily}} + \gamma L_{\text{feature}}, \tag{3}$$

where L_{attack} is the attack loss that drives the surrogate model to misclassify nodes, and $L_{\text{homophily}}$ preserves structural and feature similarity.

C. Bi-Level Optimization

To balance the conflicting goals of attack effectiveness and stealth, we employ a bi-level optimization framework. The inner loop optimizes a surrogate GNN model on the perturbed graph, while the outer loop optimizes the generator.

Homophily Preservation Loss. To explicitly maintain stealth, we define a comprehensive homophily loss L_h. This loss encourages injected nodes to have features similar to their new neighbors and ensures that new links do not violate the graph's existing homophily patterns:

$$\begin{aligned} L_{\text{h}}(G', G) = &\sum_{u \in V_{\text{inject}}} \left(1 - \cos\text{sim}(\tilde{X}'_u, X'_u)\right) \\ &+ \beta_h \sum_{(u,v) \in E_{\text{inject}}} \max(0, T - \cos\text{sim}(X'_u, X'_v)), \end{aligned} \tag{4}$$

where $\tilde{X}'_u$ is the mean feature of u's neighbors, and T is a similarity threshold.

Bi-Level Training. In the *lower-level*, we train the surrogate GNN f_s to minimize the standard cross-entropy loss L_s on the perturbed graph G'. In the *upper-level*, we optimize the generator parameters θ_g to minimize the total loss L_{total} (maximizing attack damage while minimizing stealth loss). The detailed training process is summarized in Algorithm 2.

III. Experiments and Comparisons

This section evaluates the proposed method on large-scale datasets and investigates three research aspects: (1) the method's effectiveness in performing undetectable node injection attacks on GNNs; (2) the influence of the attack budget on injection attack performance; and (3) the role of adaptive node generation and selection modules in enhancing attack performance.

A. Experimental Setup

1) Datasets: We use three graph datasets: Cora, OGB-arxiv, and Reddit. Cora is a citation network dataset where nodes represent research papers and edges indicate citation relationships. OGB-arxiv is another citation network containing paper metadata and node features based on text extraction. The Reddit dataset originates from social networks, where nodes represent posts and edges capture comment interactions. Table I summarizes the statistics.

TABLE I
Datasets statistics

Datasets	Nodes	Edges	Feature	Classes
Cora	2,708	5,429	1,443	7
OGB-arxiv	169,343	1,157,799	128	40
Reddit	232,965	11,606,919	602	41

2) Baseline Methods: We compare UNGIA with the following state-of-the-art injection attacks:

- **AFGSM [5]:** Extracts attribute vectors using GloVe embeddings, with node labels corresponding to Reddit subreddits.
- **G-NIA [20]:** A single-node injection attack that leverages optimization techniques to minimize node injections.
- **TDGIA [21]:** A directed injection attack for GNNs, misclassifying target nodes by injecting fake nodes and edges.
- **NICKI [14]:** Adds fake nodes based on inferred category feature distribution to degrade GNN classification performance.

3) Target Models: We evaluate the transferability of attacks on diverse GNN architectures:

- **GCN [10]:** Basic graph neural network extracting local structural information.
- **GraphSAGE [19]:** Generates embeddings via neighbor sampling, efficient for large-scale graphs.
- **GAT [24]:** Introduces self-attention to assign different weights to neighbor nodes.
- **RobustGCN [25]:** Enhances adversarial robustness by adding noise and regularization.
- **GCN-Prune [16]:** Prunes redundant connections to improve robustness.
- **EGNNGuard:** An advanced defense model incorporating structural and feature constraints.

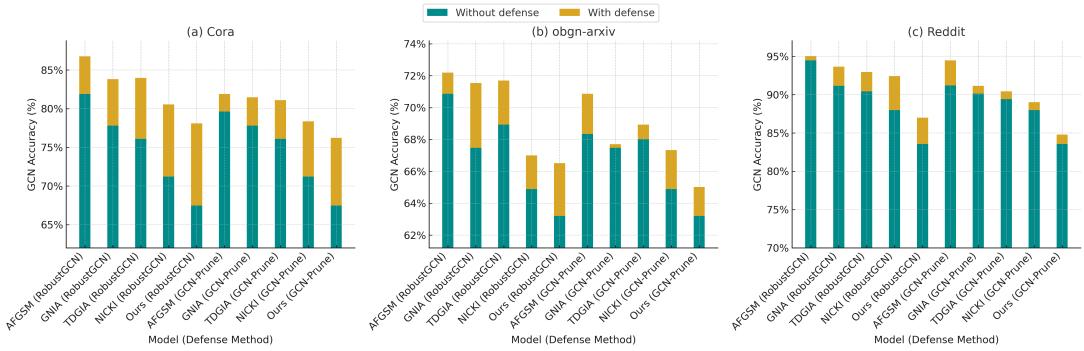


Fig. 2. Classification accuracy of different GIA models with and without GCN-Prune defense strategies. UNGIA maintains high attack efficacy (lowest accuracy bars) even under defense, while baselines lose effectiveness.

TABLE II
THE ATTACK RESULTS (CLASSIFICATION ACCURACY %) ON THREE DATASETS. LOWER IS BETTER.

Dataset	Method	GCN		GAT		GraphSAGE		RobustGCN		GCN-Prune		EGNNGuard	
		$r = 3\%$	$r = 10\%$	$r = 3\%$	$r = 10\%$	$r = 3\%$	$r = 10\%$	$r = 3\%$	$r = 10\%$	$r = 3\%$	$r = 10\%$	$r = 3\%$	$r = 10\%$
Cora	clean	81.89	81.89	81.34	81.34	80.78	80.78	86.78	86.78	79.63	79.63	81.61	81.61
	AFGSM	80.56	77.82	86.38	87.29	83.92	82.32	84.55	83.82	81.47	80.79	81.12	80.34
	GNIA	79.49	76.10	84.48	84.13	83.38	83.09	84.44	83.98	81.10	80.45	80.93	79.97
	TDGIA	74.68	71.23	74.23	74.02	74.63	73.12	83.75	83.55	78.36	77.68	77.91	76.85
	NICKI	72.09	70.03	**70.00**	73.45	75.56	70.06	84.92	83.31	78.59	77.38	**55.15**	**45.23**
	UNGIA	**71.37**	**67.47**	72.33	**70.13**	**73.16**	**69.73**	**83.09**	**81.24**	**77.22**	**75.19**	62.47	49.38
OGB-arxiv	clean	70.86	70.86	73.28	73.28	71.93	71.93	72.19	72.19	68.34	68.34	72.65	72.65
	AFGSM	69.83	67.47	71.55	70.23	71.19	69.81	71.54	70.07	67.71	66.35	67.33	65.98
	GNIA	70.24	68.93	71.88	70.57	71.33	70.02	71.70	70.22	68.03	66.72	67.89	66.54
	TDGIA	68.40	64.89	70.14	68.83	68.76	69.16	67.00	63.62	67.34	65.99	66.88	65.20
	NICKI	68.14	65.15	70.21	67.73	67.99	66.41	69.79	64.31	68.16	66.80	72.86	64.03
	UNGIA	**67.74**	**63.20**	**69.11**	**66.80**	**66.82**	**64.37**	**66.51**	**62.59**	**66.02**	**61.48**	**70.44**	**65.14**
Reddit	clean	94.48	94.48	91.29	91.29	96.43	96.43	95.03	95.03	91.20	91.20	91.57	91.57
	AFGSM	93.12	91.14	89.79	88.21	95.25	93.48	93.64	91.56	90.14	88.29	90.47	88.55
	GNIA	92.35	90.42	89.11	87.49	94.46	92.70	92.97	90.82	89.43	87.66	89.91	88.01
	TDGIA	91.76	88.98	88.62	87.02	91.18	92.21	92.42	90.35	89.01	85.72	88.62	84.91
	NICKI	92.28	88.67	90.41	86.40	92.49	81.52	90.12	85.61	89.93	87.14	87.21	86.23
	UNGIA	**88.57**	**83.02**	**88.73**	**84.80**	**88.03**	**77.60**	**86.99**	**82.35**	**84.92**	**79.96**	**83.57**	**79.42**

4) Parameter Settings: In our experiments, a two-layer GCN is used as the surrogate model and a two-layer MLP is deployed for the adaptive generator, both configured with hidden dimensions of 32. The internal iteration count, N, was fixed across all trials. The hyperparameters β and T were fine-tuned through grid search. Specifically, the T values were set to 0.5, 0.5, and 0.8 for the Cora, Reddit, and OGB-arxiv datasets, respectively. Each model undergoes 10 evaluations, and the average accuracy is reported. All computational experiments were performed on an Nvidia RTX 4090 GPU with 24GB of memory.

B. Results Analysis

1) Overall Attack Performance: Table II demonstrates UNGIA's superior attack performance. For instance, UNGIA outperforms the state-of-the-art method NICKI on the Reddit dataset by reducing the accuracy to 83.02% (GCN, $r = 10\%$), highlighting its effectiveness.

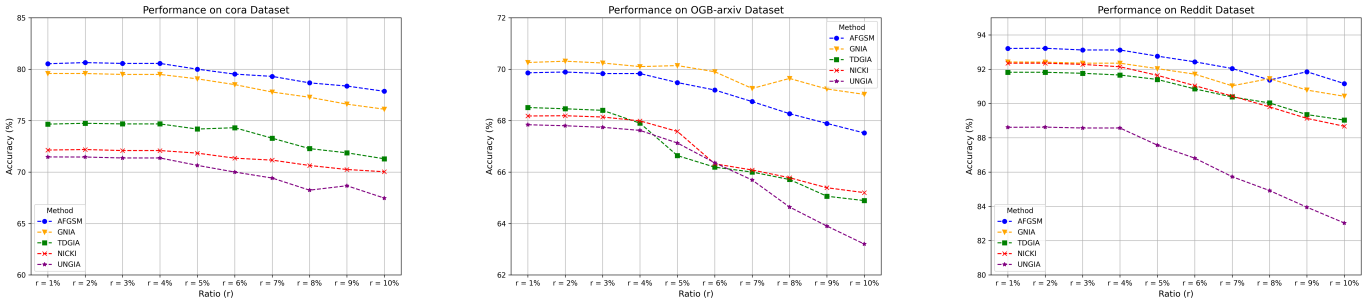
2) Attack Transferability: The transferability of UNGIA across different GNN architectures and datasets is evident from Table II. It consistently achieves greater accuracy reductions compared to baseline methods on GraphSAGE and GAT, underscoring its strong adaptability and effectiveness across diverse scenarios.

3) Attack Performance under Robust Model: Experiments using RobustGCN reveal that although the effectiveness of all attack methods decreases in adversarial settings, UNGIA consistently achieves the lowest classification accuracy. This indicates its robustness and adaptability even against advanced defense mechanisms.

4) Ablation Study: To evaluate individual component contributions, we performed an ablation study on the Reddit dataset (GCN, r=10%), summarized in Table III. Removing **clustering** slightly reduces stealth, indicating its role in structurally sound anchor selection. Disabling **adaptive features** or **link masking** further degrades unnoticeability. Crucially, removing the **homogeneity loss** (L_h) drastically reduces homophily, confirming L_h's vital role in ensuring unnoticeability.

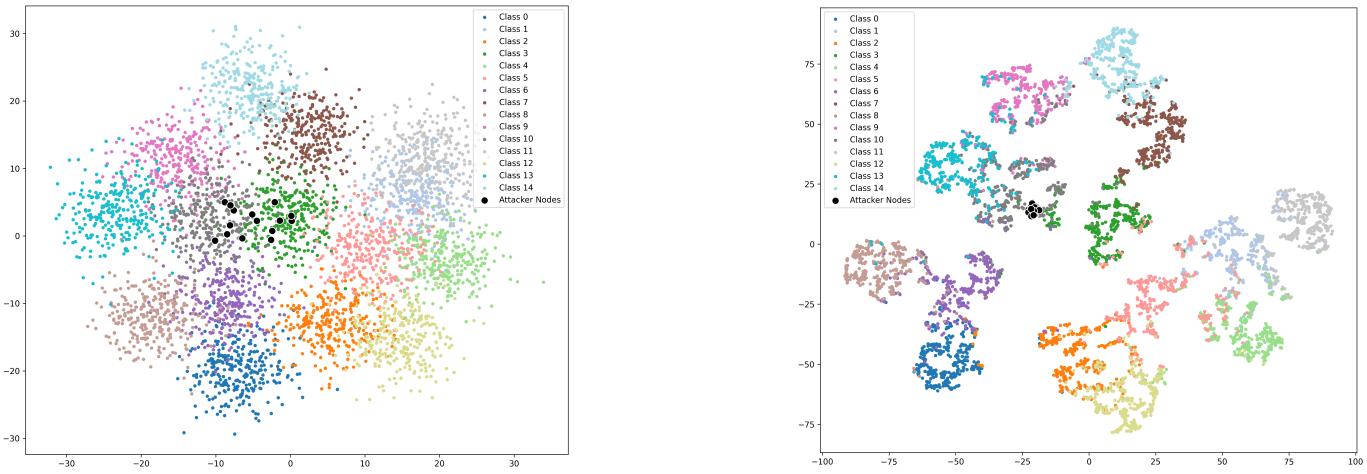
C. In-depth Analysis

1) Attack Performance under Defense Mechanisms: We further evaluated UNGIA against GCN-Prune defense. Table II illustrates the impact of defense strategies. Despite the significant reduction in attack effectiveness for other models, UNGIA



(a) Cora (b) OGB-arxiv (c) Reddit

Fig. 3. Impact of different attack budgets (r) on the performance of GCN across three datasets.



(a) Feature Distribution (b) Structural Distribution

Fig. 4. t-SNE visualization of node embeddings in the Reddit dataset. The injected fake nodes (black points) are strategically located at decision boundaries.

maintains high efficacy. As shown in Figure 2, we visualized the attack success rate comparison. UNGIA maintains a low accuracy (high success rate) even when the defense is applied, demonstrating its resilience.

2) Budget Analysis: We analyzed the scalability by varying the injection budget. As shown in Figure 3, UNGIA achieves notable improvements with increasing attack budgets. Its performance is particularly significant at a 10% budget, where it consistently outperforms baseline models across all datasets.

3) Visual Analysis of Injected Nodes: To intuitively verify the "camouflage" capability, we visualized the node embeddings using t-SNE. As shown in Figure 4, the injected fake nodes (black points) are strategically located at the decision boundaries between classes rather than being isolated outliers. This "boundary bridging" behavior makes them difficult to detect.

4) Homogeneity Analysis: Finally, we quantitatively analyzed the stealthiness. As shown in Figure 5, the homophily distribution of the graph attacked by UNGIA (d) almost perfectly overlaps with the original graph. In contrast, baselines create structural anomalies. This confirms that UNGIA preserves the macroscopic properties of the graph.

TABLE III
ABLATION STUDY ON REDDIT DATASET (GCN, $r = 10\%$)

Method Configuration	**Acc (%)** $\downarrow$	**Homophily** $\uparrow$
Clean Graph	94.48	0.85
UNGIA (Full Model)	**83.02**	**0.75**
w/o Clustering	84.55	0.72
w/o Adaptive Features	85.20	0.67
w/o Link Masking	84.78	0.61
w/o Homogeneity Loss (L_h)	82.45	0.53

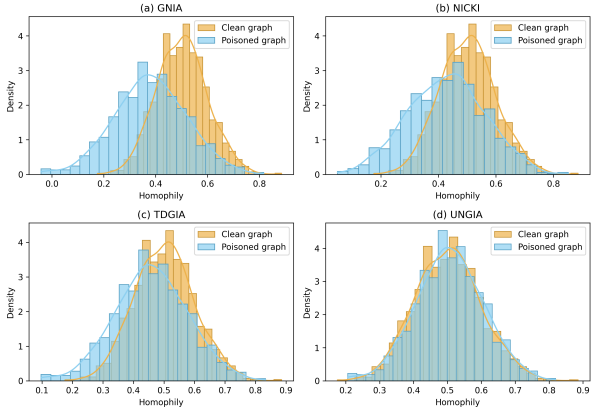


Fig. 5. Homogeneity analysis of baseline models and UNGIA. The orange area represents the clean graph, and the blue area represents the poisoned graph. UNGIA (d) shows almost perfect overlap, indicating minimal structural perturbation compared to baselines.

IV. Conclusions and Future Works

This paper's empirical verification shows that existing node injection attacks usually require a high attack budget, but can be mitigated by effective strategies. To this end, this paper proposes a new framework for implementing imperceptible graph injection attacks under budget constraints. Specifically, a node selection algorithm is designed to select representative and diverse nodes, improve attack efficiency, and assist feature learning. At the same time, an adaptive generator with imperceptibility constraints is introduced, making the injected nodes difficult to be discovered. Large-scale dataset experiments show that this method achieves good attack performance on multiple target GNN models. Future research will focus on developing defense strategies against imperceptible injection attacks.

References

[1] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems, 32(1), 4-24.

[2] Sun, Y., Wang, S., Tang, X., Hsieh, T. Y., & Honavar, V. (2020, April). Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In Proceedings of the Web Conference 2020 (pp. 673-683).

[3] Xie, Y., Liang, Y., Gong, M., Qin, A. K., Ong, Y. S., & He, T. (2022). Semisupervised graph neural networks for graph classification. IEEE Transactions on Cybernetics.

[4] Zügner, D., Borchert, O., Akbarnejad, A., & Günnemann, S. (2020). Adversarial attacks on graph neural networks: Perturbations and their patterns. ACM Transactions on Knowledge Discovery from Data (TKDD), 14(5), 1-31.

[5] Wang J, Luo M, Suya F, Li J, Yang Z, Zheng Q (2020) Scalable attack on graph data by injecting vicious nodes. Data Min Knowl Discov 34(5):1363–1389.

[6] Chen J, Wu Y, Xu X, Chen Y, Zheng H, Xuan Q (2018) Fast gradient attack on network embedding. arXiv:1809.02797

[7] Sun, Y., Wang, S., Tang, X., Hsieh, T. Y., & Honavar, V. (2020, April). Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In Proceedings of the Web Conference 2020 (pp. 673-683).

[8] Ju, M., Fan, Y., Zhang, C., & Ye, Y. (2023, June). Let graph be the go board: gradient-free node injection attack for graph neural networks via reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 37, No. 4, pp. 4383-4390).

[9] Sharma, A. K., Kukreja, R., Kharbanda, M., & Chakraborty, T. (2023). Node injection for class-specific network poisoning. Neural Networks, 166, 236-247.

[10] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

[11] Lin, X., Zhou, C., Wu, J., Yang, H., Wang, H., Cao, Y., & Wang, B. (2023). Exploratory adversarial attacks on graph neural networks for semi-supervised node classification. Pattern Recognition, 133, 109042.

[12] Chen, Y., Ye, Z., Zhao, H., & Wang, Y. (2023). Feature-Based Graph Backdoor Attack in the Node Classification Task. International Journal of Intelligent Systems, 2023(1), 5418398.

[13] Wang, X., Cheng, M., Eaton, J., Hsieh, C. J., & Wu, F. (2018). Attack graph convolutional networks by adding fake nodes. arXiv preprint arXiv:1810.10751.

[14] Sharma, A. K., Kukreja, R., Kharbanda, M., & Chakraborty, T. (2023). Node injection for class-specific network poisoning. Neural Networks, 166, 236-247.

[15] Dai, J., Zhu, W., & Luo, X. (2022). A targeted universal attack on graph convolutional network by using fake nodes. Neural Processing Letters, 54(4), 3321-3337.

[16] Wang, L., Huang, W., Zhang, M., Pan, S., Chang, X., & Su, S. W. (2022). Pruning graph neural networks by evaluating edge properties. Knowledge-Based Systems, 256, 109847.

[17] Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., & Koutra, D. (2020). Beyond homophily in graph neural networks: Current limitations and effective designs. Advances in neural information processing systems, 33, 7793-7804.

[18] Fang, J., Wen, H., Wu, J., Xuan, Q., Zheng, Z., & Chi, K. T. (2024). Gani: Global attacks on graph neural networks via imperceptible node injections. IEEE Transactions on Computational Social Systems.

[19] Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. Advances in neural information processing systems, 30.

[20] Tao, Shuchang, et al. "Single node injection attack against graph neural networks." Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2021.

[21] Zou, Xu, et al. "Tdgia: Effective injection attacks on graph neural networks." Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021.

[22] Tao, S., Cao, Q., Shen, H., Wu, Y., Hou, L., Sun, F., & Cheng, X. (2023). Adversarial camouflage for node injection attack on graphs. Information Sciences, 649, 119611.

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

[24] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

[25] Zhu, Dingyuan, et al. "Robust graph convolutional networks against adversarial attacks." Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019.