AHSG: Adversarial Attack on High-level Semantics in Graph Neural Networks

Kai Yuan   M202376878@HUST.EDU.CN
Jiahao Zhang   M202476982@HUST.EDU.CN
Xiaobing Pei[*]   XIAOBINGP@HUST.EDU.CN
Huazhong University of Science and Technology

Yidi Wang   WYIDI@FOXMAIL.COM
Great Bay University

Editors: Hung-yi Lee and Tongliang Liu

Abstract

Adversarial attacks on Graph Neural Networks aim to perturb the performance of the learner by carefully modifying the graph topology and node attributes. Existing methods achieve attack stealthiness by constraining the modification budget and differences in graph properties. However, these methods typically disrupt task-relevant primary semantics directly, which results in low defensibility and detectability of the attack. In this paper, we propose an Adversarial Attack on High-level Semantics for Graph Neural Networks (AHSG), which is a graph structure attack model that ensures the retention of primary semantics. By combining latent representations with shared primary semantics, our model retains detectable attributes and relational patterns of the original graph while leveraging more subtle changes to carry out the attack. Then we use the Projected Gradient Descent algorithm to map the latent representations with attack effects to the adversarial graph. Through experiments on robust graph deep learning models equipped with defense strategies, we demonstrate that AHSG outperforms other state-of-the-art methods in attack effectiveness. Additionally, using Contextual Stochastic Block Models to detect the attacked graph further validates that our method preserves the primary semantics of the graph.

Keywords: Adversarial attacks, Graph Neural Networks, Semantics preservation

1. Introduction

GNNs Kipf and Welling (2017); Veličković et al. (2018); Hamilton et al. (2017)have achieved significant success in graph representation learning. Due to their excellent performance, GNNs have been applied to various analytical tasks, including node classification Wu et al. (2021b), link prediction Xiong et al. (2022), and graph classification Gao et al. (2021). Recent studies have shown that similar to traditional deep neural networks, GNNs suffer from poor robustness when facing specially designed adversarial attacks. Attackers can generate graph adversarial perturbations to deceive GNNs by manipulating graph structures and node features Dai et al. (2018); Xu et al. (2019); Zügner and Günnemann (2019); Zügner et al. (2018); Liu et al. (2022), or generating new nodes and adding them to the original graph Fang et al. (2024); Chen et al. (2022). By understanding the ways in which models are vulnerable to attacks, a range of defence methods are designed that perform better when confronted with adversarial samples Zhu et al. (2019); Wu et al. (2019); Jin et al. (2020, 2021); Entezari et al. (2020).
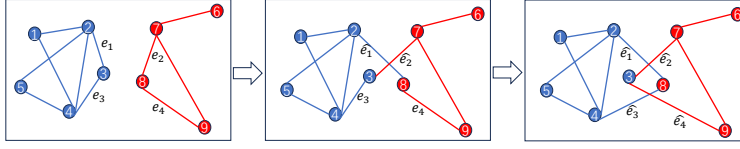
[*]Corresponding author.

Figure 1: The primary semantics of graph are disrupted under the constraints of degree distribution.

A core principle of attacking neural networks is that the attacker should preserve the primary semantics of the original data after adding perturbations. This semantic invariance ensures the practical significance of the attack. To achieve this, adversarial attacks usually require the attacker to make only minimal perturbations. Therefore, most existing research on graph adversarial attacks Dai et al. (2018); Xu et al. (2019); Zügner and Günnemann (2019); Zügner et al. (2018); Liu et al. (2022) restricts attackers to modifying a limited number of edges or nodes. However, it remains uncertain whether such attack models can effectively preserve the primary semantics. For example, real-world graphs typically have many low-degree nodes, and a small attack budget can completely remove their original neighbors. Once the edges of low-degree nodes are disrupted, their semantics are often lost. Therefore, preserving the primary semantics requires more than just managing the attack budget.

To further constrain semantic changes, existing methods introduce metrics beyond attack budget limitations. Zügner et al. (2018); Chen et al. (2022) suggest using different global graph properties as proxies for semantics, such as degree distribution and homophily. However, even under such constraints, the primary semantics can still be disrupted. Figure 1 illustrates this phenomenon with a binary classification task. Assume we select two edges $e_1 = (i, j)$ and $e_2 = (u, v)$ from the edge set of an undirected graph, and replace them with $\widehat{e_1} = (i, v)$ and $\widehat{e_2} = (u, j)$. Since $\deg(i)$, $\deg(j)$, $\deg(u)$, and $\deg(v)$ remain unchanged, this process preserves the degree distribution of the graph. We can continue to modify the edges according to this rule until the attack budget is exhausted. Finally, we successfully modifies the semantics of nodes 3 and 8 while maintaining the overall graph distribution, demonstrating the infeasibility of method Zügner et al. (2018). Therefore, neither the attack budget nor the additional constraints on graph properties can effectively preserve the primary semantics of the graph. Current research lacks a fundamental constraint aimed at maintaining primary semantics.

To address the above question, we propose a method that generates attack graph preserving primary semantics within limited attack budget. First, we need to find tools that represent semantics. Existing attack methods Dai et al. (2018); Xu et al. (2019); Zügner and Günnemann (2019); Zügner et al. (2018); Liu et al. (2022) often treat graph neural networks as black boxes, focusing only on the input and output, while ignoring the rich semantic information contained in the hidden layers. The semantics of the graph are considered to be contained in the latent representations Wu et al. (2021a), so we control the change of the latent representations to control the semantic change of the attacked graph. Although the hidden layer representations in common tasks often encapsulate the primary

semantics of the analyzed objects, task-irrelevant secondary semantics are not completely filtered out. These secondary semantics may play an inappropriately role in the model predictions. In this context, the construction of adversarial examples should lead the model to misjudge through exploring and exploiting the secondary semantics while keeping the primary semantics unchanged.

Subsequently, how can we get latent representations where primary semantics remain unchanged while secondary semantics change? Inspired by Wang et al. (2021), the entities within the same category share similar primary semantics while exhibiting diverse secondary semantics. Therefore, their linearly combined representations can preserve the majority of the consensus primary semantics, while the deviations in secondary semantics can serve as the basis for crafting attacks. We use the latent representations after linear combination as input to the deeper network layers, then use gradient ascent to find latent representations with attack effects. To generate adversarial samples, a first-order optimization algorithm is employed to map the perturbed latent representations with attack effects to the attacked graph. Finally, this secondary-semantics-targeted attack method achieves lower defensibility and detectability.

Our main contributions are as follows:

- We propose a novel adversarial attack framework AHSG based on the graph semantics, which utilizes the class-shared latent representations with similar primary semantics as the perturbation tools.

- We introduce a technique for reconstructing adversarial samples from perturbed representations by extracting deviations in secondary semantics from the linear combination of class-shared representations, while preserving the primary semantics.

- We validate AHSG on multiple acknowledged benchmark datasets, demonstrating substantial improvements under different attack settings. Semantic detection experiments confirm that AHSG preserves the primary semantics of graph data.

2. Related Work

2.1. Adversarial Attack on GNNs

In graph attacks, Nettack Zügner et al. (2018) introduces the first adversarial attack method on graphs. Meta Zügner and Günnemann (2019) treats the input graph as a hyperparameter to be learned and modifies one edge per iteration. PGD Xu et al. (2019) overcomes the challenge of attacking discrete graph structure data. Dai et al. (2018) employ reinforcement learning, gradient-based greedy algorithms, and genetic algorithms to attack GNNs in various scenarios. EpoAtk Lin et al. (2020) improves the effectiveness of the attack by bypassing potential misinformation from the maximal gradient. GraD Liu et al. (2022) generates unweighted gradients on the graph structure, unaffected by node confidence, fully utilizing the attack budget. Recent node injection attacks Zheng et al. (2021); Zou et al. (2021); Yuan et al. (2024) have been shown to be more effective than structure modification attacks. However, the issue of altering graph semantics during an attack remains underexplored. Although the attack budget is minimal relative to the entire graph, it can still lead to significant changes in the semantics of the attacked graph. Some works, such as

Nettack Zügner et al. (2018) and HAO Chen et al. (2022), maintain certain graph properties to achieve imperceptibility. However, these constraints are necessary but not sufficient to preserve semantics.

2.2. Adversarial Defense on GNNs

Some studies develop corresponding defense strategies based on the characteristics of the attacks. Jaccard Wu et al. (2019) calculates the Jaccard similarity of connected node pairs and retains only those links with high similarity. Svd Entezari et al. (2020) utilizes low-order approximations of the graph to enhance the performance of GCN against adversarial attacks. ProGNN Jin et al. (2020) jointly learns the graph structure and a robust GNN model from the perturbed graph. The SimPGCN Jin et al. (2021) framework enhances GCN robustness by effectively preserving node similarity. RGCN Zhu et al. (2019) proposes that latent representations based on Gaussian distributions can effectively absorb the impact of adversarial attacks. Recent works Yuan et al. (2024); Zhang et al. (2022) can defend against both graph modification attacks and node injection attacks. In the experiment, we use AHSG along with baseline methods to attack the aforementioned defense models.

3. Preliminary

Given an undirected attributed graph $G = (A, X)$ with n nodes, where $A \in \{0,1\}^{n \times n}$ represents the adjacency matrix, and $X \in \mathbb{R}^{n \times d}$ represents the node feature matrix. Here, d represents the feature dimension, and n represents the total number of nodes. We focus on an undirected attributed graph in this work. Formally, we denote the set of nodes as $V = \{v_i\}$ and the set of edges as $E \subseteq V \times V$. Each node v_i is associated with a corresponding node label $y_i \in Y = \{0, 1, \cdots, c-1\}$, where c is the total number of labels.

3.1. Graph Neural Networks

GNNs are specifically designed to handle graph-structured data. They leverage the message passing mechanism to aggregate information from nodes and their neighbors, thereby learning feature representations for nodes, edges, and the graph as a whole. The formula for message passing can be described as follows:

$$h_v^{(l)} = \text{UD}^{(l)} \left(h_v^{(l-1)}, h_t \right). \tag{1}$$

The representation of node v at the l-th layer, denoted as $h_v^{(l)}$, is computed by applying an update function $\text{UD}^{(l)}$ to its previous representation $h_v^{(l-1)}$ and the aggregated information h_t from its neighbors.

$$h_t = \text{AG}^{(l)} \left(\left\{ \text{MG}^{(l)} \left(h_u^{(l-1)}, h_v^{(l-1)} \right) \mid u \in \mathcal{N}(v) \right\} \right). \tag{2}$$

The aggregation function $\text{AG}^{(l)}$ collects messages generated by the message function $\text{MG}^{(l)}$, which computes the message from each neighboring node u to node v based on their respective representations $h_u^{(l-1)}$ and $h_v^{(l-1)}$. Here, $\mathcal{N}(v)$ represents the set of neighboring nodes of v. The initial node representation $h^{(0)}$ is set to the node features X. The learned representations are used as inputs for downstream tasks.

3.2. Graph Adversarial Attack

Graph adversarial attack aims to make graph neural networks produce incorrect predictions. It can be described by the following formulation:

$$
\max_{\hat{G} \in \Phi(G)} \mathcal{L}\left(f_{\theta^*}(\hat{G}(\hat{A}, \hat{X})), Y\right)
$$
$$
\text{s.t.} \theta^* = \arg\min_{\theta} \mathcal{L}\left(f_{\theta}(G'(A', X')), Y\right). \tag{3}
$$

Here, f can be any learning task function on the graph, such as node-level embedding, node-level classification, link prediction, graph-level embedding, or graph-level classification. In this paper, we primarily focus on node-level classification. $\Phi(G)$ represents the perturbation space on the original graph G. The distance between the adversarial graph and the original graph is typically measured using the attack budget or other properties. The graph $\hat{G}(\hat{A}, \hat{X})$ represents the adversarial sample.

When G' is equal to $\hat{G}$, Equation (3) represents poisoning attack. Poisoning attack occurs during model training. It attempts to influence the model's performance by injecting adversarial samples into the training dataset. On the other hand, when G' is the unmodified original G, Equation (3) represents evasion attack. Evasion attack occurs after the model is trained, meaning that the model's parameters are fixed when the attacker executes the attack. The attacker aims to generate adversarial samples specifically for the trained model. AHSG attacks the target model after its training, which categorizes it as an evasion attack.

4. Methodology

In this part, we first derive the formulation of the proposed model and then present its optimization algorithm.

4.1. Architecture of AHSG

As shown in Figure 2, AHSG can be divided into three stages.

1) Extract semantics from graph: The hidden layers of an unperturbed surrogate GNN encapsulate both the information of the node itself and that of its surrounding neighborhood after training. Therefore, we select the output of the l layer as the semantics of each node. This process can be formally described as follows:

$$
H_l = GNN_{\{1 \sim l\}}\left(A, X, W^*_{(1, \cdots, l)}\right)
$$
$$
\text{s.t.} W^*_{(1, \cdots, L)} = \arg\min_{W_{(1, \cdots, L)}} \mathcal{L}\left(f_{W_{(1, \cdots, L)}}\left(G\left(A, X\right)\right), Y\right), \tag{4}
$$

where $\mathcal{L}(.,.)$ is the loss function, commonly the cross-entropy loss. $W_{(1, \cdots, L)}$ are weight matrices. $GNN_{\{1 \sim l\}}$ refer to the first l layers of the GNN.

2) Construct adversarial latent representation that preserve the primary semantics: Wang et al. (2021) address the challenge of imbalanced datasets by extracting the feature shifts of frequent-class entities and applying them to rare-class entities. In this approach, the latent representation is assumed to be $h = P + T$, where P denotes the data
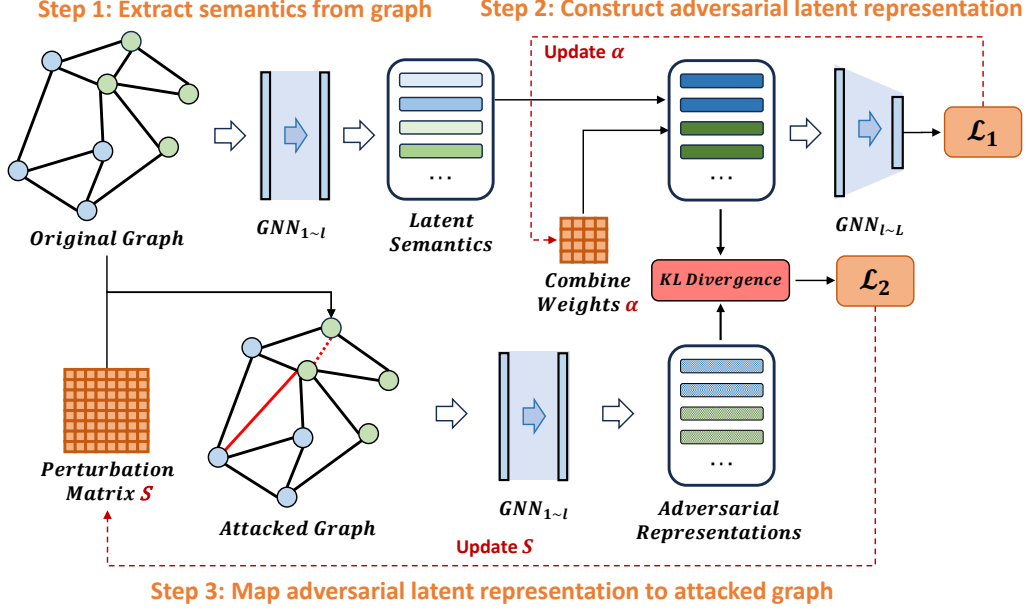
Figure 2: The framework of AHSG. It consists of three steps: extracting semantics from graph, constructing adversarial latent representations, and mapping adversarial latent representation to attacked graph.

distribution center (i.e., the similar primary semantics of same-class entities under the fixed task), and T represents the feature shift (i.e., the secondary semantics of the entities in the task). Inspired by this, we make the following assumption about the composition of the latent representation in our work:

$$h_j^i = dP(i) + eT(j), \tag{5}$$

where h_j^i denotes the latent representation vector of the j-th node in class i. It is composed of two parts: the primary semantics $dP(i)$, which is crucial for the classification task, and the secondary semantics $eT(j)$, which is less decisive. All nodes of the same class share the same primary semantic base $P(i)$, but may have different secondary semantic bases $T(j)$. Since each class contains some nodes, there exists a set where the primary semantics are preserved while the secondary semantics change. When perturbing the latent representation h_j^i of each node, we assign a weight α_k to each representation of the same class. Summing these weighted representations yields the perturbed representation $\hat{h}_j^i$.

$$\hat{h}_j^i = \frac{\sum_{k=1}^{j} \alpha_k h_k^i}{\sum_{k=1}^{j} \alpha_k}. \tag{6}$$

Using Equation (5), the representation $\hat{h}_j^i$ in Equation (6) can be derived as follows:

$$
\begin{aligned}
\hat{h}_j^i &= \frac{\sum_{k=1}^j \alpha_k h_k^i}{\sum_{k=1}^j \alpha_k} = \frac{\sum_{k=1}^j \alpha_k \left(d_k P(i) + e_k T(k) \right)}{\sum_{k=1}^j \alpha_k} \\
&= \frac{\sum_{k=1}^j \alpha_k d_k P(i) + \alpha_k e_k T(k)}{\sum_{k=1}^j \alpha_k} = \left(\frac{\sum_{k=1}^j \alpha_k d_k}{\sum_{k=1}^j \alpha_k} \right) * P(i) + \frac{\sum_{k=1}^j \alpha_k e_k T(k)}{\sum_{k=1}^j \alpha_k}.
\end{aligned}
\tag{7}
$$

As derived above, $\hat{h}_j^i$ still maintains the form $dP + eT$, where the secondary component is composed of various secondary semantics, and the primary component remains but with a modified coefficient for the primary semantics base. To ensure that the perturbed latent representation $\widehat{H}_l$ has adversarial effect on the GNN, $\widehat{H}_l$ is fed into the downstream network layers of the GNN. The optimal α is then determined based on the loss function loss_1. $[h]$ refers to arranging h in the same order as before.

$$
\mathcal{L}_1 = -\mathcal{L}(Y, GNN_{\{l \sim L\}} \left(A, \widehat{H}_l(\alpha), W^*_{(l, \cdots, L)} \right)) - \beta \operatorname{dis}(H_l, \widehat{H}_l(\alpha)),
\tag{8}
$$

where $\operatorname{dis}(.,.)$ denotes KL divergence. Now we can obtain $\widehat{H}_l(\alpha)$ as follows:

$$
\widehat{H}_l(\alpha) = \left[\hat{h}_j^i \right] = \left[\frac{\sum_{k=1}^j \alpha_k h_k^i}{\sum_{k=1}^j \alpha_k} \right].
\tag{9}
$$

To use a smaller attack budget when reconstructing adversarial samples with perturbed representations, we introduce a penalty term $\operatorname{dis}(H_l, \widehat{H}_l(\alpha))$. This term controls the similarity between the perturbed hidden layer and the original hidden layer, thereby limiting the perturbations to a certain range. Notably, it also prevents the coefficient of the primary semantics base from becoming too small. $\beta > 0$ is a hyperparameter that controls the strength of the penalty.

3) Map adversarial latent representation to adversarial sample: To construct an adversarial sample, we need to map the perturbed latent representation to the graph. Let $\hat{A}$ denote the attacked graph. We introduce a binary perturbation matrix $S \in \{0, 1\}^{N \times N}$ to encode whether an edge in G is modified. Specifically, an edge (i, j) is modified (added or removed) if $S_{ij} = S_{ji} = 1$. Otherwise, if $S_{ij} = S_{ji} = 0$, the edge (i, j) remains undisturbed. Given the adjacency matrix A, its complement matrix $\bar{A}$ is defined as $\bar{A} = 11^T - I - A$, where I is the identity matrix, and 1 is a column vector of all ones. The term $(11^T - I)$ corresponds to a fully connected graph. Using the edge perturbation matrix S and the complement matrix $\bar{A}$, Equation (10) provides the perturbed topology $\hat{A}$ of the graph A.

$$
\hat{A} = A + C \odot S, \quad C = \bar{A} - A,
\tag{10}
$$

where $\odot$ denotes the element-wise product. In the above expression, the positive entries of C indicate edges that can be added to the graph A, while the negative entries of C indicate edges that can be removed from A. Due to the difficulty in solving the problem under the binary constraint $\{0, 1\}$, this constraint is relaxed to the continuous interval $[0, 1]$. Based on

Equation (4), the latent representation is computed by using the modified graph structure $\hat{A}$, which is similar to $\widehat{H_l}$. Hence, we give the mapping loss as follows:

$$\mathcal{L}_2 = -\mathrm{dis}\left(GNN_{\{1\sim l\}}\left(\hat{A}, X, W^*_{(1,\cdots,l)}\right), \widehat{H_l}\right) \text{ s.t. } s \in S, \tag{11}$$

where $S = \{s \mid 1^T s \leq \epsilon, \ s \in [0,1]^N\}$. The function $\mathrm{dis}(.,.)$ computes the distance between matrices, commonly using KL divergence. After determining s based on $\mathcal{L}_2$, the values in s are used to sample from A, resulting in the perturbed graph $\hat{A}$.

4.2. Model Optimization

We solve AHSG step by step. In the first step, we solve for W by using a conventional gradient descent algorithm. In the second step, when solving the α problem, the number of elements in α corresponding to each node is variable because the number of same-class nodes for each node is not fixed. It is challenging to solve for unknowns of variable length in a unified manner. However, the maximum length of α is fixed, which is the total number of nodes. We set each α to the maximum length, and after each gradient descent iteration, we perform clipping according to the following formula:

$$\tau(\alpha_{i,j}) = \begin{cases} 1, & \text{if } (i = j \text{ and } y_i \text{ is unknown}) \\ \alpha_{i,j}, & \text{if } (y_i = y_j \text{ and } y_i \text{ and } y_j \text{ are known}) \\ 0, & \text{else.} \end{cases} \tag{12}$$

For the s subproblem, we scale the hard constraints $\{0, 1\}$ to $[0, 1]$. However, under the constraint $1^T s \leq \epsilon$, we still need to project the gradient-descent result of s by Equation (13). For solving μ, we use the bisection method.

$$\Pi_S(a) = \begin{cases} P_{[0,1]}[a], & \text{if } \sum P_{[0,1]}[a] \leq \epsilon \\ P_{[0,1]}[a - \mu], & \text{if } \mu > 0, \sum P_{[0,1]}[a - \mu] = \epsilon, \end{cases} \tag{13}$$

where $P_{[0,1]}(x)$ is defined as:

$$P_{[0,1]}(x) = \begin{cases} x, & \text{if } x \in [0, 1] \\ 1, & \text{if } x > 1 \\ 0, & \text{if } x < 0. \end{cases} \tag{14}$$

Sampling A using s produces the attacked graph $\hat{A}$. The overall procedure of AHSG is shown in Algorithm 1.

5. Experiment

In this section, we validate the effectiveness of AHSG through comprehensive experiments.

Algorithm 1 The procedure of AHSG

Input: Original graph $G = (A, X)$
Parameter: $E, \lambda, \beta, T, \eta_t, K, h$
Output: $\hat{A}$

1: Train GNN to obtain $W^*_{(1,2)}$.
2: Calculate H_l via Equation(4).
3: **for** $e \leftarrow 1$ to E **do**
4:     Calculate $\widehat{H_l}$ via Equation(9).
5:     Gradient descent:
    $\alpha(e) = \alpha(e-1) - \lambda \nabla \mathcal{L}_1 (\alpha(e-1))$.
6:     Calculate α via Equation(12).
7: **end for**
8: **for** $t \leftarrow 1$ to T **do**
9:     Gradient descent:
    $a(t) = s(t-1) - \eta_t \nabla \mathcal{L}_2(s(t-1))$
10:    Call projection operation in Equation(13).
11: **end for**
12: **for** $k \leftarrow 1$ to K **do**
13:    Generate p of the same size as s that follows a uniform distribution.
14:    Draw binary vector $d^{(k)}$ following
   $d_i^{(k)} = \begin{cases} 1, & \text{if } s_i > p_i \\ 0, & \text{if } s_i \leq p_i \end{cases}$
15: **end for**
16: Choose a vector s^* from $d^{(k)}$ which yields the smallest loss$_2$ under $1^T s \leq \epsilon$.
17: Calculate $\hat{A}$ via Equation(10).

5.1. Experimental Settings

We evaluated AHSG on five well-known datasets: Cora, Citeseer, Cora-ML, Ogbn-arxiv, and Amazon-Photo. Cora, Citeseer, and Cora-ML are citation networks where nodes represent papers with bag-of-words features and edges represent citations. Ogbn-arxiv is a large-scale citation network from arXiv with nodes representing papers and edges representing citations. Amazon-Photo is a co-purchase graph where nodes represent products and edges indicate frequent co-purchases. The statistical results are shown in Table 1.

The baselines include Random, Meta Zügner and Günnemann (2019), GradAM Dai et al. (2018), PGD Xu et al. (2019), EpoAtk Lin et al. (2020), GraD Liu et al. (2022), EBAG Chanda et al. (2025). We choose GCN as the surrogate model, where the total number of layers L in the neural network is 2, and l is 1. We conduct comparative experiments with advanced graph attack and defense methods. For Svd and Jaccard, we adjust the order of high-order approximations and the threshold for removing low-similarity links to achieve optimal defense performance. Other attack methods use default parameters.

In our method, the default settings are as follows: the number of iterations E for perturbation is set to 300, with a learning rate λ of 0.3. The regularization coefficient β is 0.2. For reconstruction, the number of iterations T is 300, and the dynamic learning rate η_t is $\frac{q}{\sqrt{t+1}}$, where t denotes the iteration count and q is set to 20. The number of selections K for the probability matrix is 20. When using a two-layer GCN as the victim model, the

Datasets	Nodes	Links	Features	Classes	Binary
Cora	2708	5278	1433	7	Y
Citeseer	3327	4552	3703	6	Y
Cora-ML	2995	8158	2879	7	N
Ogbn-arxiv	169343	1166243	128	40	N
Amazon-Photo	7650	143663	745	8	Y

Table 1: Statistics of datasets. The last column indicates whether the dataset has binary features.

dimension h of the first layer is 128, and the second layer's dimension corresponds to the number of classes.

5.2. Attack Performance on GNNs

Except for ProGNN, other GNNs use the evasion attack setting. ProGNN performs both graph structure purification and parameter learning simultaneously, so the attack occurs before training. The attack results on different GNNs by modifying 10% of the edges are shown in Table 2. We observe that all baseline methods lead to a performance drop in the victim models. AHSG outperforms all other attackers on all datasets. Furthermore, AHSG achieves the best attack performance on GCN among different GNN models, since GCN is the corresponding surrogate model, indicating a white-box attack scenario. Furthermore, even against the five defensive GNNs, AHSG still demonstrates strong attack performance, highlighting its robust generalization capability derived from the surrogate model.

Dataset	Method	Clean	Random	Meta	GradAM	PGD	EpoAtk	GraD	EBAG	AHSG
Cora	GCN	0.823	0.809	0.809	0.716	0.708	0.685	0.673	0.657	**0.646**
	Jaccard	0.786	0.784	0.736	0.712	0.759	0.693	0.703	0.712	**0.676**
	SVD	0.729	0.711	0.665	0.673	0.686	0.689	0.715	0.705	**0.648**
	ProGNN	0.809	0.794	0.659	0.737	0.703	0.673	0.665	0.698	**0.654**
	SimPGCN	0.790	0.782	0.784	0.703	0.728	0.703	0.695	0.713	**0.682**
	RGCN	0.801	0.748	0.753	0.690	0.678	0.671	0.664	0.675	**0.651**
Citeseer	GCN	0.666	0.655	0.650	0.578	0.593	0.549	0.542	0.558	**0.528**
	Jaccard	0.665	0.632	0.645	0.617	0.623	0.591	0.622	0.604	**0.576**
	SVD	0.601	0.593	0.573	0.588	0.565	0.589	0.574	0.573	**0.558**
	ProGNN	0.683	0.625	0.531	0.566	0.603	0.597	0.582	0.577	**0.530**
	SimPGCN	0.656	0.646	0.648	0.585	0.615	0.613	0.602	0.613	**0.581**
	RGCN	0.610	0.579	0.569	0.576	0.573	0.559	0.552	0.559	**0.534**
Cora-ML	GCN	0.859	0.851	0.827	0.743	0.791	0.741	0.738	0.742	**0.733**
	Jaccard	0.860	0.839	0.807	0.770	0.828	0.801	0.813	0.784	**0.762**
	SVD	0.820	0.799	0.794	0.759	0.816	0.763	0.808	0.797	**0.757**
	ProGNN	0.840	0.805	0.791	0.749	0.823	0.783	0.769	0.778	**0.735**
	SimPGCN	0.846	0.836	0.817	0.743	0.818	0.774	0.758	0.760	**0.740**
	RGCN	0.859	0.842	0.831	0.754	0.828	0.785	0.771	0.809	**0.755**

Table 2: Accuracy of GNNs with 10% edge modifications. The best result in each row is highlighted in bold.

5.3. Attack Performance w.r.t Attack Budget

Table 3 presents the accuracy on the test sets on five datasets as the proportion of perturbed edges increases from 5% to 20%. For example, in the case of the Cora dataset, as anticipated,

Dataset	Attack Ratio	Random	Meta	GradAM	PGD	EpoAtk	GraD	EBAG	AHSG
Cora	5%	0.814	0.814	0.765	0.756	0.731	0.725	0.712	**0.698**
	10%	0.809	0.809	0.716	0.708	0.685	0.673	0.657	**0.646**
	15%	0.795	0.805	0.685	0.681	0.634	*OOM*	0.625	**0.615**
	20%	0.787	0.798	0.652	0.645	0.607	*OOM*	0.593	**0.585**
Citeseer	5%	0.657	0.651	0.608	0.610	0.585	0.578	0.585	**0.565**
	10%	0.655	0.650	0.578	0.593	0.549	0.542	0.558	**0.528**
	15%	0.653	0.631	0.539	0.564	0.523	*OOM*	0.512	**0.495**
	20%	0.641	0.628	0.507	0.529	0.501	*OOM*	0.495	**0.487**
Cora-ML	5%	0.853	0.841	0.791	0.843	0.795	0.789	0.792	**0.776**
	10%	0.851	0.827	0.743	0.791	0.741	0.738	0.742	**0.733**
	15%	0.846	0.821	0.721	0.751	0.711	*OOM*	0.723	**0.700**
	20%	0.837	0.809	0.695	0.708	0.698	*OOM*	0.701	**0.685**
Ogbn-arxiv	5%	0.715	0.703	0.686	0.695	0.674	0.681	0.669	**0.651**
	10%	0.703	0.697	0.653	0.663	0.653	*OOM*	0.643	**0.623**
	15%	0.686	0.693	0.632	0.641	0.627	*OOM*	0.627	**0.597**
	20%	0.673	0.685	0.614	0.611	0.602	*OOM*	0.593	**0.563**
Amazon-Photo	5%	0.797	0.783	0.774	0.776	0.751	0.757	0.765	**0.732**
	10%	0.794	0.775	0.751	0.742	0.714	0.701	0.713	**0.692**
	15%	0.786	0.751	0.734	0.721	0.683	0.672	*OOM*	**0.653**
	20%	0.775	0.732	0.709	0.714	0.664	0.653	*OOM*	**0.624**

Table 3: Accuracy of GCN under different attack budgets. The best result in each row is highlighted in bold. OOM represents Out of Memory.

the test accuracy consistently decreases with a higher number of perturbed edges, although the rate of decline slows down. Moreover, the proposed AHSG achieves the best attack performance across all perturbation ratios. Similar conclusions are drawn for the Citeseer, Cora-ML, Ogbn-arxiv, and Amazon-Photo datasets.

5.4. Semantic Detection

In the fields of computer vision (CV) and natural language processing (NLP), humans can effectively perform semantic checks. However, in the domain of graph, it is challenging for humans to inspect the semantics of large-scale graphs. To address this, Gosch et al. (2023) attempt to define semantic boundaries by introducing a reference classifier g to represent changes in semantic content. The reference classifier g can be derived from knowledge about the data generation process. According to the data generation process of CSBMs, Gosch et al. (2023) use a Bayes classifier as g.

Data generation process: synthetic graphs with analytically tractable distributions are generated using the Contextual Stochastic Block Models (CSBMs). It defines the edge probability p between same-class nodes and q between different-class nodes. Node features are extracted using a Gaussian model. Sampling from CSBMs can be described as an iterative process for node $i \in n$:

1. Sample the label $y_i \sim \text{Ber}(1/2)$ (Bernoulli distribution).

2. Sample the feature vector $x_i \mid y_i \sim \mathcal{N}((2y_i - 1)\mu, \sigma I)$, where $\mu \in \mathbb{R}^d$ and $\sigma \in \mathbb{R}$.

3. For all $j \in n$, if $y_i = y_j$, then sample $A_{j,i} \sim \text{Ber}(p)$; otherwise, sample $A_{j,i} \sim \text{Ber}(q)$ and set $A_{i,j} = A_{j,i}$.

We denote this as $(X, A, y) \sim \text{CSBM}(n, p, q, \mu, \sigma^2)$.

We reclassify the nodes in the attacked graph $G(X', A')$, which is generated by Equation (15).

$$G(X', A') = \text{attack}(G(X, A)). \tag{15}$$

As shown in Equation (16), if the reference classifier g gives the same prediction for node v in both the original and attacked graph, and the prediction is correct, then Gosch et al. (2023) consider that the primary semantics of node v have not been altered by the attack.

$$g(X', A')_v = g(X, A)_v = y_v,$$
$$(X, A, y) \sim \text{CSBM}(n, p, q, \mu, \sigma^2). \tag{16}$$

In Equation (17), the reference classifier g consists of a feature-based Bayesian classifier and a structure-based Bayesian classifier. The structure-based Bayesian classifier considers four scenarios: whether nodes of the same class are connected and whether nodes of different classes are connected.

$$g(X, A)_v = \arg\max_y \left(Bayes(X, A, v, y) \right),$$
$$Bayes(X, A, v, y) = log(p\left(X_v | y_v \right) +$$
$$log(\prod_{i=0}^{n} p^{A[i,v](1-|y_i-y_v|)} (1-p)^{(1-A[i,v])(1-|y_i-y_v|)}$$
$$q^{A[i,v](|y_i-y_v|)} (1-q)^{(1-A[i,v])(|y_i-y_v|)}). \tag{17}$$

The number of nodes with consistent classification results before and after the attack can be regarded as the degree of graph semantics preservation. We denote this by Bayes_maintain, as shown in Equation (18), where $\mathbb{I}$ is an indicator function.

$$\text{Beyes\_maintain} = \frac{\sum_{i=0}^{n} \mathbb{I}_{g(X',A')_v=g(X,A)_v=y_v}}{n}. \tag{18}$$

As shown in Table 4, the Bayes reference classifier achieves 92.4% accuracy on clean graphs, validating its effectiveness as a semantic proxy. Comparison with other attack methods reveals that AHSG alters the Bayesian classification results the least, thereby maintaining the majority of the primary semantics in the classification task while achieving the best attack performance.

Attack Method	Clean	Meta	GradAM	PGD	EpoAtk	GraD	EBAG	AHSG
GCN	0.724	0.653	0.632	0.607	0.597	0.593	0.614	0.586
Beyes_maintain	0.924	0.810	0.802	0.832	0.774	0.783	0.804	0.903

Table 4: Accuracy of GCN and the degree of primary semantic preservation under different attack methods.

Note that in CSBMs, We use the original default settings. The number of generated nodes n is 1000. Node feature mean μ is calculated as $\frac{M\sigma}{2\sqrt{d}}$, with variance $\sigma = 1$, $d = \frac{n}{(\ln(n))^2} = 21$, and $M = 0.5$. Smaller value of M causes the reference classifier to rely more

on structural information. Since AHSG focuses on structural attacks, it is essential for the reference classifier to analyze the semantics within the structure. The edge probability p between same-class nodes is 0.6326%, while the edge probability q between different-class nodes is 0.1481%.

5.5. Ablation Study

To further validate the effectiveness of AHSG, we conduct an ablation study to analyze the impact of its various components. Since AHSG consists of a perturbation module, a reconstruction module, and a regularization module, we performed ablation study by removing one of these modules to assess their influence on AHSG's performance. The regularization module is discussed in the hyperparameter analysis. The specific combinations for the ablation study are as follows: AHSG-rec refers to the version where we retain the perturbation module of AHSG but choose random connections during the reconstruction process. AHSG-hid refers to the version where we retain the reconstruction module of AHSG but apply random perturbations during the perturbation process. As shown in Table 5, the best attack performance is achieved only when both perturbation and reconstruction steps are present simultaneously. This result is reasonable. Without the guidance of the latent representation, the reconstruction step cannot capture the correct perturbation information. Conversely, without the reconstruction step, the perturbed latent representation cannot be transformed into an effective attack graph.

Modules	Cora	Citeseer	Cora-ML
Clean	0.823	0.666	0.859
AHSG-rec	0.809	0.655	0.852
AHSG-hid	0.804	0.650	0.842
AHSG	0.698	0.565	0.776

Table 5: Accuracy of GCN with 10% edge modifications on different modules and datasets.

5.6. Parameter Analysis

Finally, we performed a hyperparameter analysis on the regularization term coefficient β and the hidden layer dimension h. AHSG achieves optimal performance when β is around 0.1, as this value effectively balances the contributions of the attack loss and the regularization term. Large β restricts the range of perturbation in the hidden layers, preventing the attack from fully utilizing the attack budget. Conversely, small β can result in an excessively large distance between the perturbed representation and the original representation, making it difficult to reconstruct the perturbed representation within a given attack budget. When changing the hidden layer dimension h of the surrogate model, AHSG's performance shows considerable fluctuation. Particularly, when the hidden layer dimension is small, the attack effectiveness decreases significantly. Because the hidden layers with a small dimension can not adequately capture the information of each node (including both primary and secondary semantics) for AHSG to utilize. However, it is worth noting that an attack graph generated using a surrogate model with a large hidden layer dimension can still be effective when applied to a target model with a small hidden layer dimension.
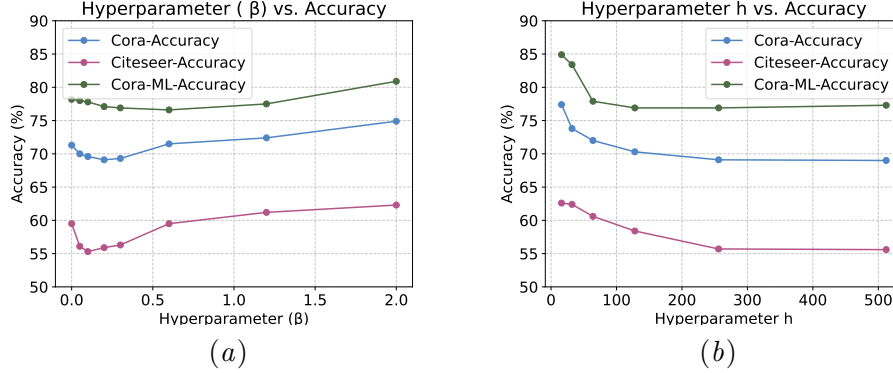
Figure 3: Hyperparameter analysis on the regularization term coefficient β and the hidden layer dimension h.

6. Conclusion

We investigate graph structure attacks on GNNs under the evasion setting and propose AHSG, which preserves primary semantics in hidden layers to prevent significant semantic disruption. By exploiting the similarity of primary semantics among nodes with the same class, AHSG constructs representations that cause GNN failure while maintaining semantic consistency. Adversarial graphs are generated using the PGD algorithm, with a regularization term added to ensure semantic invariance. Experiments on various datasets show AHSG's superior attack performance against multiple GNNs, including defense models. Semantic detection confirms that AHSG effectively preserves task-relevant primary semantics.

References

Dibaloke Chanda, Saba Heidari Gheshlaghi, and Nasim Yahya Soltani. Explainability-based adversarial attack on graphs through edge perturbation. *Knowledge-Based Systems*, 310: 112895, 2025.

Yongqiang Chen, Han Yang, Yonggang Zhang, Kaili Ma, Tongliang Liu, Bo Han, and James Cheng. Understanding and improving graph injection attack by promoting unnoticeability. In *International Conference on Learning Representations*, 2022.

Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR, 2018.

Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th international conference on web search and data mining*, pages 169–177, 2020.

Junyuan Fang, Haixian Wen, Jiajing Wu, Qi Xuan, Zibin Zheng, and K Tse Chi. Gani: Global attacks on graph neural networks via imperceptible node injections. *IEEE Transactions on Computational Social Systems*, 2024.

Jianliang Gao, Jun Gao, Xiaoting Ying, Mingming Lu, and Jianxin Wang. Higher-order interaction goes neural: A substructure assembling graph attention network for graph classification. *IEEE Transactions on Knowledge and Data Engineering*, 35(2):1594–1608, 2021.

Lukas Gosch, Daniel Sturm, Simon Geisler, and Stephan Günnemann. Revisiting robustness in graph machine learning. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

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

Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. Node similarity preserving graph convolutional networks. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 148–156, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017. URL https://arxiv.org/abs/1609.02907.

Xixun Lin, Chuan Zhou, Hong Yang, Jia Wu, Haibo Wang, Yanan Cao, and Bin Wang. Exploratory adversarial attacks on graph neural networks. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1136–1141. IEEE, 2020.

Zihan Liu, Yun Luo, Lirong Wu, Zicheng Liu, and Stan Z Li. Towards reasonable budget allocation in untargeted graph structure attacks via gradient debias. In *Advances in Neural Information Processing Systems*, 2022.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018. URL https://arxiv.org/abs/1710.10903.

Jianfeng Wang, Thomas Lukasiewicz, Xiaolin Hu, Jianfei Cai, and Zhenghua Xu. Rsg: A simple but effective module for learning imbalanced datasets. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3784–3793, 2021.

Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples on graph data: Deep insights into attack and defense, 2019. URL https://arxiv.org/abs/1903.01610.

Likang Wu, Zhi Li, Hongke Zhao, Qi Liu, Jun Wang, Mengdi Zhang, and Enhong Chen. Learning the implicit semantic representation on graph-structured data. In *Database Systems for Advanced Applications: 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11–14, 2021, Proceedings, Part I 26*, pages 3–19. Springer, 2021a.

Man Wu, Shirui Pan, and Xingquan Zhu. Openwgl: open-world graph learning for unseen class node classification. *Knowledge and Information Systems*, 63(9):2405–2430, 2021b.

Bo Xiong, Shichao Zhu, Nico Potyka, Shirui Pan, Chuan Zhou, and Steffen Staab. Pseudo-riemannian graph convolutional networks, 2022. URL https://arxiv.org/abs/2106.03134.

Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective, 2019. URL https://arxiv.org/abs/1906.04214.

Xiangchi Yuan, Chunhui Zhang, Yijun Tian, Yanfang Ye, and Chuxu Zhang. Mitigating emergent robustness degradation while scaling graph learning. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Chunhui Zhang, Yijun Tian, Mingxuan Ju, Zheyuan Liu, Yanfang Ye, Nitesh Chawla, and Chuxu Zhang. Chasing all-round graph representation robustness: Model, training, and optimization. In *The eleventh international conference on learning representations*, 2022.

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

Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1399–1407, 2019.

Xu Zou, Qinkai Zheng, Yuxiao Dong, Xinyu Guan, Evgeny Kharlamov, Jialiang Lu, and Jie Tang. Tdgia: Effective injection attacks on graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2461–2471, 2021.

Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2847–2856, 2018.