

Momentum gradient-based untargeted poisoning attack on hypergraph neural networks

Chen, Y; Picek, S.; Ye, Z.; Wang, Z.; Zhao, H.

2025, Article / Letter to editor (Neurocomputing, 634, (2025), pp. 1-14, article 129835)

Doi link to publisher: <https://doi.org/10.1016/j.neucom.2025.129835>

Version of the following full text: Publisher's version

Published under the terms of article 25fa of the Dutch copyright act. Please follow this link for the

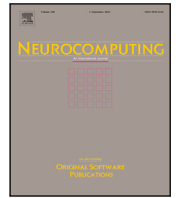
Terms of Use: <https://repository.ubn.ru.nl/page/termsfuse>

Downloaded from: <https://hdl.handle.net/2066/317304>

Download date: 2026-02-28

Note:

To cite this publication please use the final published version (if applicable).



Momentum gradient-based untargeted poisoning attack on hypergraph neural networks

Yang Chen^{a,b,c,*}, Stjepan Picek^d, Zhonglin Ye^{b,c}, Zhaoyang Wang^{b,c}, Haixing Zhao^e

^a School of Computer Science and Technology, Shandong Technology and Business University, Yantai, Shandong, 264005, China

^b School of Computer Science, Qinghai Normal University, Xining, Qinghai, 810000, China

^c The State Key Laboratory of Tibetan Intelligent Information Processing and Application, Xining, 810000, Qinghai, China

^d Radboud University, Houtlaan 4, 6525 XZ, Netherlands

^e Qinghai Minzu University, Xining, Qinghai, 810000, China

ARTICLE INFO

Communicated by M. Bianchini

Keywords:

Hypergraph neural networks

Untargeted poisoning attack

Momentum gradient

Feature attack

ABSTRACT

Hypergraph Neural Networks (HGNNs) have been successfully applied in various hypergraph-related tasks due to their excellent higher-order representation capabilities. Unfortunately, recent works have shown deep learning models vulnerable to diverse attacks. Most studies of attacks on graphs have focused on Graph Neural Networks (GNNs), and the study of attacks on HGNNs remains largely unexplored. In this paper, we try to bridge this gap. We design a new untargeted poisoning attack for HGNNs, MGHGA, which focuses on modifying node features. We consider the process of HGNNs training and use a surrogate model to implement the attack before hypergraph modeling. Precisely, MGHGA consists of two parts: feature selection and feature modification. We use a momentum gradient mechanism to choose the attack node features in the feature selection module. In the feature modification module, we use two feature generation approaches (direct modification and sign gradient) to enable MGHGA to be employed on discrete and continuous datasets. We conduct extensive experiments on seven benchmark datasets to validate the attack performance of MGHGA in the node and the visual object classification tasks. The results show that MGHGA improves performance by an average of 2% compared to the baselines.

1. Introduction

Graph Neural Networks (GNNs) are widely used in tasks such as graph classification [1,2], node classification [3,4], and link prediction [5] due to their efficient learning ability and generalization capability. Graphs provide a useful way to represent pairwise connections between objects in real-world networks. However, they may not fully capture the complex higher-order relationships between objects. Indeed, when dealing with multimodal data, GNNs cannot efficiently learn all the information between objects, resulting in missing information and lower efficiency [6,7]. For example, researchers are abstracted as nodes in the scientist collaboration network, and edges represent the paper collaboration relationship. The common graph cannot represent the situation where multiple researchers work together on a paper [8]. However, a hypergraph can represent this complex relationship. Specifically, the hyperedge (an edge in a hypergraph is called a hyperedge) represents the collaborations for a paper, and a hyperedge connecting K nodes means that K researchers collaboratively work on a paper.

Thus, the hypergraph has an advantage over the common graph in modeling complex relationships. Hypergraph Neural Networks (HGNNs) based on the hypergraph also outperform GNNs in many tasks [9,10].

In recent years, many works have demonstrated the vulnerability of GNNs to various security and privacy threats like evasion attacks, poisoning attacks, backdoor attacks, and membership inference attacks, resulting in degraded performance [11–13]. Many attacks on graphs aim to undermine its performance by adding small perturbations to the graph [14]. Depending on the attack's goal, the attacks can be classified as targeted and untargeted attacks. The targeted and untargeted attacks are focused on the classification of the target node and all nodes, respectively. The targeted attack makes the target nodes misclassified with the target label [15]. The untargeted attack misclassifies the nodes with no labeling restrictions [16]. Since the targeted attack usually attacks users with higher privileges, they could be detected by defense models and could be challenging to implement in real-world attacks [17]. As such, many works are based on the untargeted attack [18]. HGNNs, as an extension of hypergraph deep learning on graph data, also show vulnerability to adversarial attacks [19].

* Corresponding author at: School of Computer Science and Technology, Shandong Technology and Business University, Yantai, Shandong, 264005, China.
E-mail address: chenyang2753@sdtbu.edu.cn (Y. Chen).

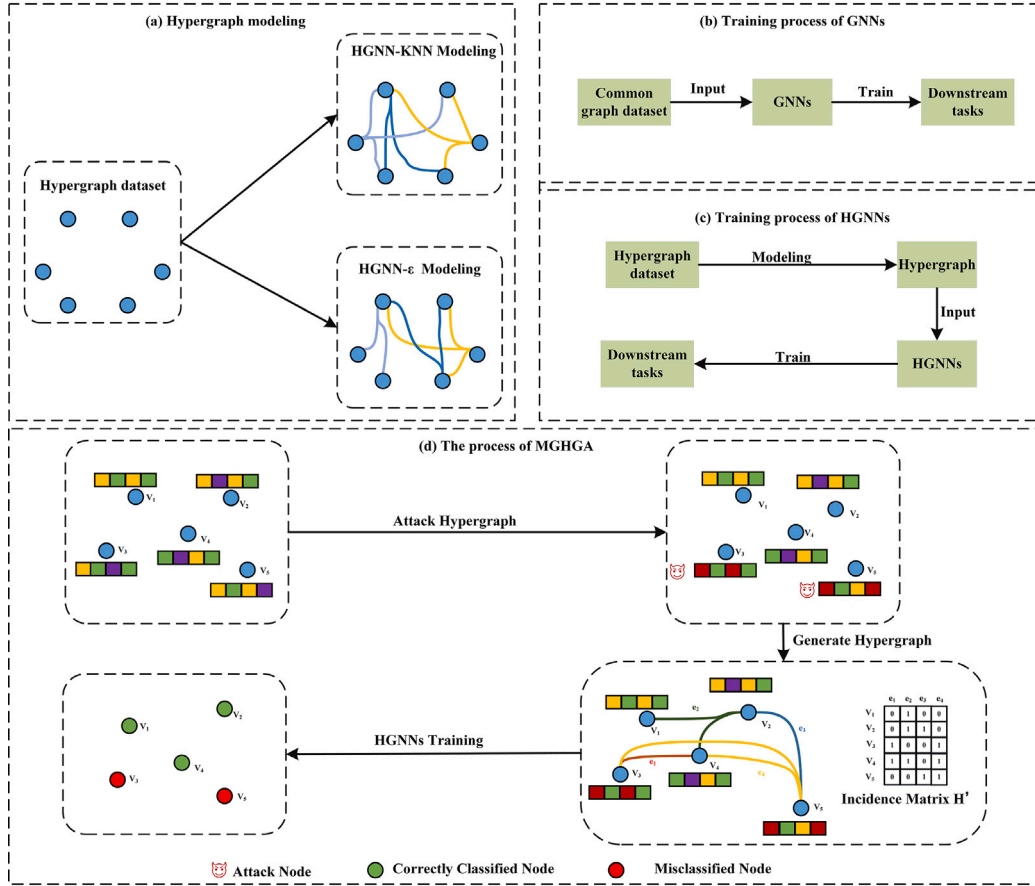


Fig. 1. (a) shows two hypergraph modeling approaches. (b) and (c) represent the GNNs' and HGNNs' training processes, respectively. (d) shows the MGHGA process.

Almost all of the current works on attacks on graphs focus on GNNs and ignore the security of HGNNs [20,21], making HGNNs difficult to apply in practice. For example, HGNNs are often used in disease detection tasks (e.g., Alzheimer's disease) [22], the attacker add malicious noise to the pathology data can make it hard for doctors to understand the patient's condition and make wrong decisions. There are some differences between HGNNs and GNNs when dealing with the data. For example, GNNs can only process simple graph data, while HGNNs can process complex and high-dimensional data. Due to the specificity of hypergraph data, we summarize two main challenges in HGNNs attacks from the perspective of hypergraph data:

1. **Challenge 1 — Lack of structure.** Unlike common graph datasets, there is no association between nodes in most hypergraph datasets, and the hypergraph structure (adjacency relations) can be obtained through various modeling techniques [23].
2. **Challenge 2 — Continuous Features.** Common graph datasets consist of data with discrete features. Hypergraph datasets can be applied in tasks such as graph visualization [24] and image classification [25], to name a few examples. Since many hypergraph datasets have continuous features, the majority of attacks on GNNs cannot be directly applied to HGNNs.

HGNNs differ from GNNs in the convolution operation. Specifically, graph convolution is defined based on edges between nodes, while hypergraph convolution is defined based on hyperedges between nodes [26,27]. Hypergraph convolution is more complex than graph convolution, which leads to a more difficult implementation of attacks in HGNNs. To the best of our knowledge, only HyperAttack [19] has made a preliminary exploration of attacks on HGNNs. HyperAttack uses the gradient to modify the hyperedges. However, the HyperAttack implementation assumes that the hypergraph dataset has been

modeled to obtain the hypergraph structure (the original hypergraph dataset is unstructured). In other words, HyperAttack implements the attack in the hypergraph with a fixed structure. Fig. 1(a) shows the hypergraph structure H obtained using two distance-based HGNNs (HGNN-KNN, HGNN- ϵ [28]). Notice that the hypergraph structure H is obtained differently under different approaches. In addition, different settings of HGNNs' parameters generate different hypergraph structures in practice. Therefore, HyperAttack has attack instability as it exhibits different performances in different structural hypergraphs generated for the same dataset. Then, the defender can add or delete useless hyperedges to reduce the attack's efficiency [29].

To address the above challenges, in this paper, we propose an attack that is more applicable to HGNNs, namely MGHGA. Unlike HyperAttack, MGHGA is an untargeted attack (the attack aims to decrease the accuracy of HGNNs at all nodes), allowing more diverse attack scenarios. Figs. 1 (b) and (c) represent the training process of GNNs and HGNNs, respectively. Observe that modeling the dataset is a prerequisite for training HGNNs, where as GNNs do not require this step. The previous attacks directly modify the graph's structure [30], and HyperAttack is no exception. To address the challenge 1, we take a new perspective of attacking features before hypergraph modeling. MGHGA uses momentum gradients to guide the attacker to modify the features. We consider the scenarios of attacking discrete and continuous features separately to address the challenge 2. In discrete datasets, MGHGA directly inverts the feature values from 0 to 1 and vice versa. In continuous datasets, we update the features using a sign gradient strategy. Finally, the new hypergraph structure is obtained by modeling the perturbed hypergraph dataset and feeding it into the HGNNs to verify the validity of the MGHGA. The above process is shown in Fig. 1(d). To highlight the novelty of our attack, Table 1 shows the differences between HyperAttack and MGHGA.

We summarize the contributions of this paper as follows:

Table 1
Comparison between HyperAttack and MGHGA.

Attack	Target of attack	Phase of attack	Downstream task
HyperAttack	Targeted attack	After hypergraph modeling	Node classification
MGHGA	Untargeted attack	Before hypergraph modeling	Node and Visual object classification

- We propose the first untargeted poisoning attack against HGNNs. Our attack, MGHGA, considers the training characteristics of HGNNs and implements the attack before hypergraph modeling.
- We propose a momentum gradient method to guide the attacker in modifying the node's features so that our attack can be applied to both discrete and continuous hypergraph datasets.
- Extensive experiments on seven datasets verified that MGHGA could effectively reduce the effectiveness of HGNNs and outperform baseline methods in node and visual object classification tasks.
- We conduct an ablation study showing that our attack works under different parameters and combinations of surrogate and victim models.

The rest of this work is organized as follows. Section 3 introduces the background information about the HGNNs and untargeted attacks. Section 4 introduces MGHGA in detail, including feature selection and feature modification elements. Section 5 presents the evaluated datasets, parameters, and results. In Section 2, we review the works on hypergraph learning and attacks on graphs. Finally, we conclude our work and propose future research directions in Section 6.

2. Related work

2.1. Hypergraph learning

The flexibility and capability of hypergraph learning to model complex higher-order data correlations have garnered increasing attention in recent years [31,32]. Hypergraph learning usually consists of constructing hypergraphs and designing hypergraph learning methods [33, 34].

Constructing hypergraphs. There are four types of methods for constructing hypergraphs: distance-based, representation-based, attribute-based, and network-based. Specifically, Huang et al. [28] proposed a nearest neighbor construction method that aims to find adjacent vertices in the feature space and construct a hyperedge to connect them. Wang et al. [35] proposed a representation-based hyperedge construction mechanism that exploits the correlation between feature vectors to establish node connections. Huang et al. [36] proposed a generation method applicable to attribute hypergraphs, which uses attribute information to construct hypergraphs. Fang et al. [37] used user friendship and mobility information to construct hypergraphs in the location social networks.

Hypergraph learning methods. Hypergraph learning can be divided into spectral analysis methods, neural network methods, and other methods according to their implementations [38]. Feng et al. [6] first proposed the hypergraph neural network, which extends the spectral approach of graph convolutional neural networks to the hypergraph and designs hypergraph convolution operations. Yadati et al. [39] proposed the HyperGCN, which solves the problem of processing semi-supervised classification on the hypergraph. Huang et al. [31] proposed MultiHGNN, which learns multimodal hypergraph data and uses hypergraph modeling for each modality to accomplish downstream tasks. In order to address the problem that HGNNs can lead to excessive smoothing, the literature [40] proposed a Distance Enhanced Hypergraph Learning (DEHL) method for dynamic node classification. The DEHL temporally-adaptive pre-training component generates time-aware representations of nodes. Then, a two-channel convolution module is utilized to construct higher-order relations. Song et al. [41] proposed a Contrast Hypergraph Neural Network (CHGNN), which utilizes self-supervised contrast learning techniques from both labeled and unlabeled data.

2.2. Attacks on graph neural networks

Attacks on graph neural networks can be classified into different types, mainly attack type, target, knowledge, and capability. Attacks can be classified into three categories based on their type: **the topology attack, the feature attack, and the hybrid attack** [42–44]. In the topology attack, the attacker focuses on modifying the graph topology, which is a common attack method, e.g., FGA [42], Mettack [45], RL-S2V [29], and HyperAttack [19]. The node feature modification is another common attack method, where the attacker focuses on modifying the features of the nodes, e.g., GANI [43]. In Nettack [46] and IG-Attack [44], the attacker uses the graph topology and node feature attacks to degrade GNNs' accuracy. Shi et al. [47] proposed an H3NI attack against HGNNs, which uses genetic algorithms and predefined budget models to implement hypergraph injection attacks.

Based on the target of the attack, we can classify the attacks into **targeted and untargeted attacks** [15,48]. Dai et al. [15] proposed a targeted universal attack against GNNs, where the attacker's goal is to misclassify some of the test nodes into the attacker-specified labels. Fang et al. [43] injected fake nodes with malicious information into the graph, making the GNN perform poorly on the test nodes. Cao et al. [49] proposed a multi-view ingestion candidate edge learning attack (GSA-MPCEL), which accomplishes the attack by modifying the structure and centers on designing a multi-view ingestion candidate edge learning module and a guided attack loss module to guide the attack.

According to the attacker's knowledge, we can divide attacks into three categories: **white-box attack, gray-box attack, and black-box attack**. In a white-box attack, the attacker has knowledge about the GNNs model and datasets [45]. In a gray-box attack, the attacker only has some knowledge, e.g., knowing the parameters of GNNs but not the prediction results of nodes [46]. In black-box attacks, the attacker does not know the model architecture, parameters, and training data and can only obtain a small amount of model feedback [29]. The attack improved the success rate by querying the node information and using batch normalization to enhance the dissimilarity of node representations. Hussain et al. [30] proposed a gray-box attack where the attacker can access the labels of nodes and disrupt the fairness of node classification by injecting adversarial links. Ju et al. [50] proposed a black-box attack method using a reinforcement learning framework, where the attacker is not using a surrogate model to query model parameters or training labels.

In this paper, we study HGNNs adversarial attacks. Since there are some structural differences between GNNs and HGNNs, these differences also influence them to exhibit different characteristics under adversarial attacks. In GNNs, adversarial attacks mainly involve interfering with node features or edge attributes to lead to the output of erroneous results. In HGNNs, adversarial attacks may involve more complex interference patterns, such as interfering with properties of hyperedges or node combinations, which may lead to more challenges. Fig. 2 shows the HGNNs and GNNs feature attack process. Specifically, when HGNNs attack features, the structure of the hypergraph changes with the modification of the features. Fig. 2(a) shows that the features of the nodes of the perturbation graph and the hyperedge associations are changed, e.g., nodes V_3 and V_5 exist within the same hyperedge in the perturbation graph. GNNs attack only the features change, the graph structure does not change. Fig. 2(b) shows that only the features of attacking nodes V_3 and V_5 are changed and node neighborhoods are unchanged in the perturbation graph. We have proposed MGHGA, which affects the relationship between hyperedges and nodes by perturbing node attributes thus leading to the degradation of HGNNs performance.

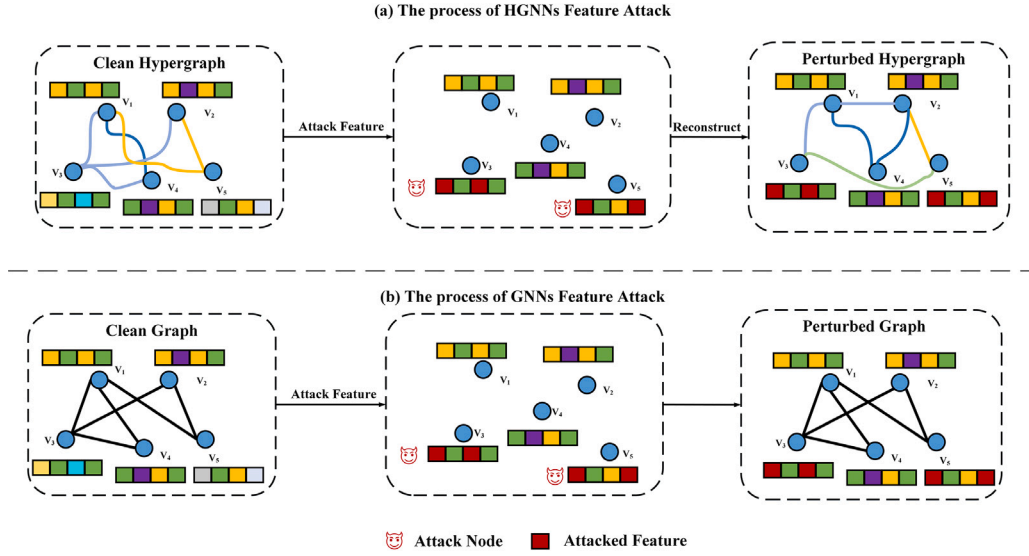


Fig. 2. HGNNs and GNNs feature attack process.

3. Preliminaries

3.1. Hypergraph neural network

Given a hypergraph dataset $D = (V, X)$, where $V = \{v_1, v_2, \dots, v_{|V|}\}$ represents the set of nodes, $X \in \mathbb{R}^{|V| \times d}$ denotes the node feature matrix, and d denotes the dimensions of the feature. Let $\mathcal{G} = (V, E, W)$ be a hypergraph, where $E = \{e_1, e_2, \dots, e_{|E|}\}$ represents the set of hyperedges, $\text{diag}(W) = [w(e_1), w(e_2), \dots, w(e_{|E|})]$ is the diagonal matrix of the hyperedge weights, and $w(e)$ is the weight of the hyperedge. The correlation matrix $H \in \{0, 1\}^{|V| \times |E|}$ is used to represent the structure of the hypergraph, $H(v, e) = 1$ if the node v is inside the hyperedge e , and $H(v, e) = 0$ otherwise, which can be expressed as follows:

$$H(v, e) = \begin{cases} 1, & \text{if } v \in e. \\ 0, & \text{if } v \notin e. \end{cases} \quad (1)$$

The classification task is one of the common tasks in HGNNs [31]. Here, we set the downstream task of HGNNs as node and visual object classification. The hypergraph convolutional network learns node representations, transforms, and propagates information to accomplish downstream tasks. A hypergraph convolutional layer can be represented as [6]:

$$X^{(l+1)} = \sigma \left(D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2} X^{(l)} \Theta^{(l)} \right), \quad (2)$$

where $X^{(l)}$ denotes the node representation of the hypergraph at l th layer. $\sigma(\cdot)$ denotes the nonlinear activation function, W is the weight matrix of the hyperedges, D_e represents the diagonal matrix of the hyperedge degree (hyperedge degree is the number of nodes contained in the hyperedge), D_v is the diagonal matrix representing the node degree (node degree is the number of hyperedges containing the node), and $\Theta^{(l)}$ denotes the l th layer training parameters.

We set the number of layers in HGNNs to two as in related works, e.g., [6], representing it as:

$$Z = f(H, X) = \text{softmax} \left(\hat{H} \text{ReLU} \left(\hat{H} X \Theta^{(1)} \right) \Theta^{(2)} \right), \quad (3)$$

where $\hat{H} = D_v^{-1/2} H W D_e^{-1} H^T D_v^{-1/2}$. $\Theta^{(1)}$ and $\Theta^{(2)}$ are the training parameters of the first and second layers, respectively.

In the training phase, our goal is to optimize $\Theta = (\Theta^{(1)}, \Theta^{(2)})$ to obtain the optimal classifier $f_{\Theta^*}(H, X)$:

$$\min_{\Theta} L_{\text{model}} = - \sum_{u \in V_L} Y_u \ln(Z_{u,\cdot}), \quad (4)$$

where $Z_{u,\cdot}$ denotes the set of predicted labeling probabilities for node u , Y_u denotes the true label of node u , V_L denotes the training set of node, and the predicted label of node u is denoted as

$$\hat{Y}_u = \arg \max(Z_{u,\cdot}). \quad (5)$$

3.2. Untargeted poisoning attack

The untargeted attack aims to reduce the global classification performance of HGNNs. Given a budget Δ that represents the number of times the attacker modifies the feature entities, the untargeted attack can be expressed as:

$$\begin{aligned} & \arg \max_{X'} \sum_{v \in V_T} \mathbb{I}(Y_v \neq C_v), \\ & \text{s.t. } C = \arg \max_{C'} f_{\Theta^*}(H, X'), \\ & \Theta^* = \arg \min_{\Theta} L_{\text{model}}(H, X'), \|X - X'\| \leq \Delta, \end{aligned} \quad (6)$$

where C is the set of node prediction labels, and V_T denotes the test set of nodes. $\mathbb{I}(x)$ is the indicator function. If x is true, $\mathbb{I}(x)$ returns 1, otherwise $\mathbb{I}(x)$ returns 0. The primary rationale for Eq. (6) is that a model with large training error will likely generalize poorly on the test set.

3.3. Threat model

Depending on whether the attack occurs before or after the HGNNs training, it can be categorized into the evasion attack or poisoning attack [51–53]. An evasion attack is performed on the trained HGNNs, and the attacker cannot modify the parameters or structure of the HGNNs [54]. The poisoning attack is performed before the HGNNs are trained, and the attacker inserts perturbations in the training data to interfere with the training process of the HGNNs [55]. In the evasion attack, the attacker aims to modify the links or features of the test nodes, causing the nodes to be misclassified [56]. The poisoning attack has the advantage of being a more powerful attack and easier to implement than the evasion attack. For example, in an e-commerce recommendation system, graph deep learning models are used to predict the recommendation of a target item based on the sales records of existing goods. The attacker tries to modify the graph information to reduce the recommendation rate of the target product, thus affecting the sales of the product and resulting in lower revenue for the merchants. It is easier for an attacker to obtain information about sold goods (training nodes) than obtaining information about

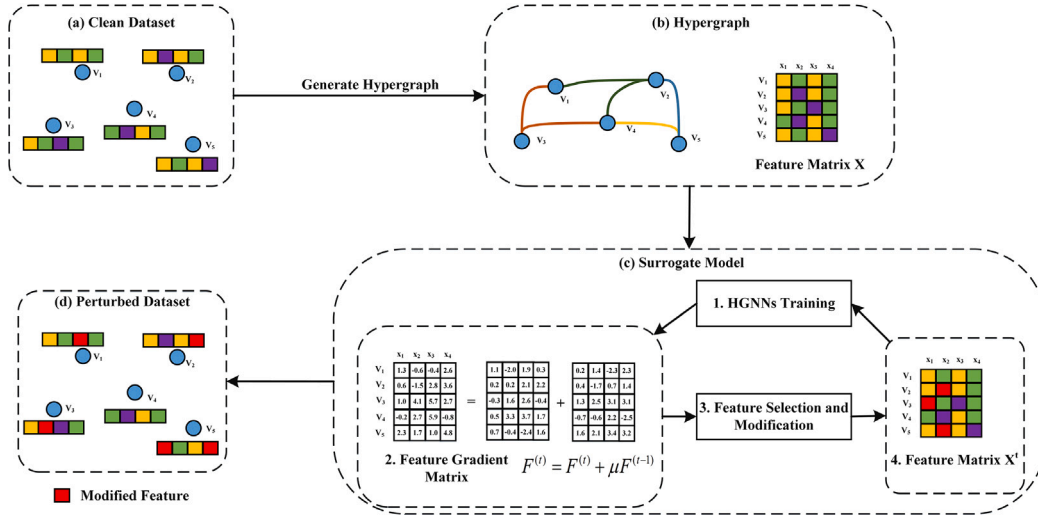


Fig. 3. The pipeline of MGHGA. The goal of MGHGA is to generate perturbed datasets with malicious information. The red color represents the features of the nodes under attack. In module (c), the surrogate model is first trained on the HGNNs several times (1. HGNNs Training) and the feature gradient matrix F' is computed by the momentum gradient algorithm (2. Feature Gradient Matrix). Then the features are found and modified according to the gradient modification rule (3. Feature Selection and Modification), and finally the new feature matrix X' is obtained (4. Feature Matrix). The goal of the surrogate model is that the features modified at each iteration increase the training loss of the surrogate HGNNs. The optimal perturbation feature is obtained when the attack is completed.

recommended goods (test nodes), so the poisoning attack is easier to implement than an evasion attack. Moreover, with the poisoning attack, the graph depth model behavior can be directly changed by modifying the training data, and thus, the attack effect may be more powerful and lasting. In contrast, the evasion attack is usually targeted at specific test samples, and the impact of the attack may be relatively short-lived.

We focus on mounting a poisoning attack against HGNNs, where the core idea is to use the surrogate model to obtain the perturbed dataset before the HGNNs are trained. The use of surrogate models is a common strategy to implement the poisoning attack. An attacker uses a surrogate model to simulate the behavior of the original model in order to trick the system into thinking that the data provided by the attacker is legitimate. By constructing a surrogate model similar to the original model, an attacker can more easily generate deceptive input data that makes it difficult for the system to recognize the truth. In addition, the use of a surrogate model can help an attacker better understand the workings of the system and the structure of the model, thus allowing for a more targeted attack. As we evaluate the robustness of HGNNs, we also set the surrogate model to HGNNs. In addition, our attack is a white-box attack, which requires constant feedback information (e.g., gradient or node prediction labels) from surrogate HGNNs.

4. Momentum gradient hypergraph attack (MGHGA)

In this section, we detail the MGHGA components. The MGHGA pipeline is shown in Fig. 3. MGHGA addresses two challenges in hypergraph attacks, as already discussed in Section 1. First, many hypergraph datasets do not have a correlation relationship among nodes. Attacking the hypergraph structure does not guarantee the stability of the attack due to the different hypergraph structures generated for the same hypergraph dataset under different modeling approaches. To solve this problem, we use the surrogate model to attack the nodes' features before the hypergraph modeling, as shown in Fig. 3(c). Second, hypergraph datasets can be classified as discrete or continuous based on feature attributes. To improve the MGHGA, we designed two methods to update the features.

4.1. Feature selection

We use the gradient of the feature matrix in the surrogate model to select features denoted as

$$F_{i,j} = \frac{\partial L_{model}}{\partial X_{i,j}}. \quad (7)$$

Here, $F \in \mathbb{R}^{|V| \times d}$ is the gradient matrix of features. A larger gradient of a feature indicates that this feature has a greater impact on the optimization of HGNNs, and thus, modifying the feature could have a large impact on HGNNs. However, previous studies have shown that using a greedy approach to directly modify the feature with the largest gradient makes the attack susceptible to falling into the optimized local optimum and easily overfitting to the attack model, which can diminish the generalizability of the generated adversarial samples [46].

To address the above problem, we propose a momentum gradient hypergraph attack. The momentum method is a technique used to accelerate the gradient descent algorithm by accumulating velocity vectors along the gradient direction of the loss function during the iterations [57]. Accumulating previous gradients helps the model avoid falling into local optimum. We apply the momentum method to generate malicious features.

First, the momentum gradient matrix of the features is computed as:

$$\begin{cases} F^0 = 0. \\ F^t = \mu F^{t-1} + \frac{\partial L_{model}}{\partial X^{t-1}}. \end{cases} \quad (8)$$

Here, μ represents the momentum decay factor, and $X^{(t)}$ denotes the perturbation feature matrix¹ after t iterations, $X^{(0)} = X$.

MGHGA selects the feature of the maximum absolute value of the momentum gradient matrix:

$$m_{i,j}^t = \arg \max |F^t|, \quad (9)$$

where $m_{i,j}^t$ represents the absolute maximum value of gradient at t iterations, i and j represent the node and feature indices, respectively.

4.2. Feature modification

We modify only one feature at each iteration and stop the attack when the number of modifications reaches the budget Δ . Here, we consider discrete and continuous feature modifications, respectively.

¹ The perturbation feature matrix is the feature matrix after the attack. When calculating the features that are important at moment t , the gradient is first calculated based on the perturbation matrix obtained from the previous round of attacks, and then the features that are attacked at moment t are selected.

Table 2

Statistics of seven datasets. We use three discrete datasets (Cora, Citeseer and DBLP-CA) and two continuous dataset (Cora-ML and Pubmed) in the node classification task. Visual object datasets are usually continuous datasets (NTU, ModelNet40), and we use two datasets to validate the attack's effectiveness.

Datasets	Nodes	Features	Classes	Discrete
Cora	2485	1433	7	Y
Cora-ML	2810	2879	7	N
Citeseer	3327	3703	6	Y
Pubmed	19717	500	3	N
DBLP-CA	41302	1425	6	Y

Datasets	Nodes	Features1	Features2	Classes	Discrete
NTU	2012	4096	2048	67	N
ModelNet40	12311	4096	2048	40	N

Discrete Features. We use a direct inversion mechanism to modify the features for discrete features, where the feature values are only 1 or 0. At t iterations, feature deletion or addition can be expressed as

$$X'_{i,j} = 1 - X_{i,j}^{t-1}, \quad \text{s.t.} \quad \|X^t - X\| \leq \Delta. \quad (10)$$

Continuous Features. For the attack with continuous features, the researchers use the gradient sign mechanism to generate perturbations, and the results indicate that this mechanism can achieve a high-efficiency attack, as demonstrated in [58]. Inspired by this, we use a gradient sign mechanism to update the continuous features. Our approach involves introducing perturbations along the opposite direction of the normal sample gradient to maximize the target model's training loss error, lowering its classification confidence and increasing the probability of inter-class confusion, ultimately leading to misclassification. Specifically, the update feature matrix can be expressed as:

$$X'_{i,j} = X_{i,j}^{t-1} + \eta \text{sign}(F_{i,j}^t), \quad \text{s.t.} \quad \|X^t - X\| \leq \Delta, \quad (11)$$

where η is the constraint factor and $\text{sign}(x)$ is denoted as the gradient symbol. $\text{sign}(x) = 1$ when $x > 0$, otherwise $\text{sign}(x) = -1$. At the end of the iteration, the anomalous features are filtered to avoid the newly generated features not being within the bounds of the original features.

$$\begin{cases} X'_{i,j} = \arg \min X, & \text{If } X'_{i,j} < \arg \min X. \\ X'_{i,j} = \arg \max X, & \text{If } X'_{i,j} > \arg \max X. \end{cases} \quad (12)$$

X and X' denote the original and perturbed hypergraph feature sets, respectively. $\arg \min(X)$ and $\arg \max(X)$ denote the minimum and maximum values of the original hypergraph feature set, respectively. We reset the features outside the lower and upper bounds of the original features, which ensures the imperceptibility of the perturbations and prevents them from being detected by simple defense mechanisms (e.g., outlier detection). Note that filtering anomalous features in MGHGA does not consume the budget. Since multiple modifications of the same feature cause a waste of the allocated budget, we attack at most once for both continuous and discrete features.

4.3. Training loss

MGHGA aims to generate adversarial samples by modifying features. Specifically, the attacker's goal is to maximize the loss of the adversarial samples, which can be achieved by minimizing the model's prediction accuracy on the adversarial samples. The attacker makes the loss function larger by adjusting the inputs, which in turn leads to a decrease in the performance of the model. Therefore, the loss function of MGHGA is defined as follows:

$$L = \arg \max L_{\text{model}}(H, X'), \quad \text{s.t.} \quad \|X - X'\| \leq \Delta. \quad (13)$$

Algorithm 1 MGHGA

Input: Hypergraph dataset $D = (V, X)$, momentum decay factor u , constraint factor ϵ , budget Δ

Output: Perturbation hypergraph dataset $D' = (V, X')$

```

1: Initialization: Modeling hypergraph  $\mathcal{G} = (V, E, W)$ , HGNN surrogate model  $f_{\Theta^*}(H, X)$ , perturbation gradient matrix  $F^0$ 
2: while  $\|X^t - X\| \leq \Delta$  do
3:   Calculate the gradient of  $i$  iterations through Eq. (7)
4:   Calculate the  $i$  iteration momentum gradient through Eq. (8)
5:   Select the  $i$  iteration features through Eq. (9)
6:   if  $X$  is the discrete feature then
7:     Update gradient matrix through Eq. (10)
8:   else  $\triangleright X$  is the continuous feature
9:     Update gradient matrix through Eq. (11)
10:  end if
11: end while
12: if  $X$  is the continuous feature then
13:   Filter features through Eq. (12)
14: end if

```

4.4. Algorithm

The pseudocode of MGHGA is given in Algorithm 1.

Complexity Analysis. MGHGA uses HGNN as a pre-trained model, and HGNN includes forward and backward propagation in training with a complexity $\mathcal{O}(t\|H\| \cdot \|X\|)$, where t denotes the number of HGNN training steps. Then, MGHGA calculates the gradient of the feature matrix, and the complexity is $\mathcal{O}(d|V|)$. Updating and filtering features are basic operations with low complexity and are ignored here. The complexity of modifying features is $\mathcal{O}(Td|V|)$, where T denotes the number of attack iterations. In summary, the overall complexity of MGHGA is $\mathcal{O}(t\|H\| \cdot \|X\| + Td|V|)$.

5. Experimental results

5.1. Datasets

Recent works have shown that HGNNs exhibit excellent performance on node classification and visual object classification tasks [6, 31], which are the most common practical applications of HGNNs. Therefore, our work focuses on these two tasks. To illustrate the performance of MGHGA, experiments are carried out on seven datasets. We performed node classification tasks on Cora, Cora-ML, Citeseer [59], Pubmed and DBLP-CA [60] datasets. The visual object classification task is performed on two multi-feature and continuous datasets: the National Taiwan University 3D model (NTU) [61] and Princeton ModelNet40 (ModelNet40) [62]. The dataset information is summarized in Table 2. Hypergraphs are obtained by modeling hypergraph datasets. However, NTU and ModelNet40 are without adjacencies, and their adjacencies are obtained using the hypergraph construction methods. To ensure the consistency of the adjacency relations of each dataset, we do not use the original adjacency of the Cora, Cora-ML, Citeseer, Pubmed and DBLP-CA datasets but instead utilize commonly used construction methods to generate hypergraph structures in the experiments [23].

5.2. Baselines

Since MGHGA is the first work on the untargeted poisoning attack in HGNNs, there are no related works to consider directly. HyperAttack, which is most relevant to our work, is a targeted attack that cannot be used as a comparison. Due to the specificity of the hypergraph structure, it is difficult to migrate the GNN attacks to HGNNs directly. Thus, we use the following approaches for comparison.

Random Attack (Random): The conclusion of the work on common graph attacks shows that the Random Attack can degrade the performance of GNNs [46]. In this paper, we attack the features randomly. For discrete or continuous features, we randomly select features to attack within the budget. In other words, it is not every feature that is attacked. Specifically, the features to be attacked are randomly changed from 0 to 1 and from 1 to 0 in the discrete dataset. In the continuous dataset, the features to be attacked are randomly modified. It should be noted that the modified features are within the range of the original features.

Node Degree Attack (NDA): Previous works have shown that attacking nodes with maximum node degree degrades the performance of the GNNs [63]. Extending to HGNNs, NDA attacks nodes with maximum node degree. Note that features are modified in the same way as for the Random Attack.

Fast Gradient Attack (FGA): Fast Gradient Attack is a common gradient attack on graph data [42]. We extend it to hypergraphs. In each attack, we choose the feature with the largest absolute value of the gradient to attack.

Fast Gradient Attack-Node Degree (FGA-D): In this setting, we add a constraint for the FGA to attack the node with the larger degree. Note that the FGA and FGA-D modify the discrete features similarly to MGHGA.

EpoAtk-E: We extend the GNNs gradient attack EpoAtk to attack HGNNs [64], called EpoAtk-E. EpoAtk was originally used to attack the structure, where EpoAtk-E only uses the generation and evaluation modules to modify the features, and no longer uses the reorganization module to modify the hypergraph structure. Specifically, EpoAtk-E generates a set of candidate perturbations by calculating the feature gradients, and then selects the optimal set of candidates as the optimal perturbations through the evaluation function.

MGHGA-D: MGHGA-D is an extension of MGHGA, and MGHGA-D is obtained from MGHGA with the same constraint as FGA-D.

Note that FGA-D and MGHGA-D are attacks with constraints, i.e., they attack nodes with high node degree.

5.3. Parameter setting and metrics

Parameters. In our experiments, the hypergraph is generated using two distance-based generation methods: HGNN-KNN and HGNN- ϵ [23], where K and ϵ are set to 10 and 0.5, respectively. The correlation matrix H is set to a binary matrix. HGNNs are set to two layers, the feature dimension of the hidden layer is set to 64, and dropout is applied to avoid overfitting. In the training process, the training count is set to 300, and the learning rate of the Adam optimizer is 0.001. The ratio of the training set to the test set is 0.2 and 0.8, respectively. The constraint factor η in the discrete features dataset is set to X_{avg} . The constraint factor η is set to $X_{avg} = \frac{\sum(X)}{|V|d}$ for the continuous features dataset. The budget $\Delta = \lambda|V|$, where λ is the budget factor set to 0.05 by default. The decay factor μ is 0.8. FGA-D and MGHGA-D attack nodes with top 1% node degree by default. The number of EpoAtk-E candidate sets is set to 10. The victim model and the target model are the same by default, where the victim model is the model used by the user, and it is the pre-trained model used by the surrogate model. The experiments are conducted on a PC with an Intel(R) Xeon(R) Gold 5118 processor and two NVIDIA GeForce GTX 4080Ti GPUs.

Metrics. We use the Classification Success Rate (CSR) to measure the attack's effectiveness, the mathematical expression is shown in Eq. (14). The CSR metric usually indicates the percentage of nodes that are correctly classified by HGNNs. When performing an attack, we want to make more nodes misclassified by adversarial perturbations. If the value of the CSR is small, it means that the model performs poorly against the samples and the percentage of misclassified nodes is high. This is exactly what the attacker hopes to achieve. Therefore, a small

value of the metric indicates that the HGNNs model performs poorly and the attack is effective.

$$CSR = \frac{\sum_{v \in V_{test}} \mathbb{I}(Y_v = C_v)}{|V_{test}|}, \text{ s.t. } C = \arg \max_{f_{\theta^*}} (H, X'). \quad (14)$$

Where $|V_{test}|$ is the set of test nodes.

In addition, we used MAE and LogLoss metrics in the ablation experiments to comprehensively analyze the performance of MGHGA. In graph deep learning, MAE and LogLoss are metrics for evaluating the prediction accuracy of a model. However, in the case of adversarial attacks, the goal of the attacker is to degrade the performance of the model. Therefore, higher values of MAE and LogLoss indicate better attack performance.

MAE measures the average absolute difference between the predicted value and the true value. If the attack is successful, the model's predictions will deviate more from the true value and the MAE will increase. Therefore, the higher the MAE, the more effective the attack is.

$$MAE = \frac{1}{|V_{test}|} \sum_{i=1}^{|V_{test}|} |\hat{Y}_u - Y_u|. \quad (15)$$

LogLoss is usually used in classification problems to measure the difference between the predicted probability distribution and the true distribution. The larger the LogLoss, the greater the difference between the model's predicted probability distribution and the true distribution, and the more effective the attack.

$$LogLoss = -\frac{1}{|V_{test}|} \sum_{i=1}^{|V_{test}|} \sum_{j=1}^{|Y|} Y_{ij} \log \hat{Y}_{ij}. \quad (16)$$

where $|Y|$ denotes the number of label classes. $\hat{Y}_{ij}$ denotes the probability that node i belongs to label j .

5.4. MGHGA attack performance

Table 3 summarizes the CSR of the seven datasets under the attacks. We can see the difference in CSR for the same dataset under different hypergraph modeling methods. Taking Cora as an example, the CSR of HGNN-KNN and HGNN- ϵ are 59.31% and 57.19%, respectively. HGNN-KNN performs better than HGNN- ϵ , and the results are the same in other datasets. This suggests that different hypergraph structures can have a significant impact on downstream tasks. Since the hypergraph structure is modeled by node features, it is more effective to attack node features. In particular, to verify the performance of MGHGA under large-scale datasets, we conducted experiments on Pubmed, DBLP-CA, and ModelNet40. The results show that MGHGA achieves the best performance in larger scale datasets as well.

The performance of HGNN-KNN and HGNN- ϵ decreases under all attacks, which indicates that attacking node features can effectively degrade the performance of HGNNs. Specifically, we observe that MGHGA achieves the best performance under all the datasets. For example, using HGNN-KNN as the victim model, the CSR of the discrete dataset Cora are 58.65%, 58.47%, 57.63%, 57.21% and 55.33% for Random, NDA, FGA, EpoAtk-E and MGHGA, respectively. The lower CSR indicates that the attack causes more damage to HGNNs. Therefore, MGHGA achieves the best efficiency in comparing the advanced attacks. The results are the same in other datasets, especially for the NTU and ModelNet40 continuous features datasets, which shows that our proposed method applies not only to discrete features datasets but also to continuous features datasets.

In all datasets, HGNN-KNN has a higher CSR than HGNN- ϵ after the attack, which is due to the different modeling approaches. The KNN method can select nodes more accurately as neighbors that are more similar to the original node's attributes. In addition, the number of neighbors chosen by the KNN method is fixed and not affected by the value of ϵ , which helps to improve the stability and robustness of the

Table 3

Comparison of CSR (%) of several attacks. The lower the CSR, the better the attack performance. For all cases, the best results are given in bold style. The results are an average of ten runs.

Datasets	Model	Clean	Random	NDA	FGA-D	FGA	EpoAtk-E	MGHGA-D	MGHGA
Cora	HGNN-KNN	59.31 $\pm$ 0.3	58.65 $\pm$ 0.9	58.47 $\pm$ 0.5	58.24 $\pm$ 0.8	57.63 $\pm$ 1.1	57.21 $\pm$ 1.3	58.13 $\pm$ 0.9	55.33 $\pm$ 1.8
	HGNN- ϵ	57.19 $\pm$ 0.2	56.97 $\pm$ 1.1	56.73 $\pm$ 0.6	56.46 $\pm$ 0.7	55.10 $\pm$ 1.0	54.20 $\pm$ 0.7	54.94 $\pm$ 0.9	53.51 $\pm$ 1.5
Cora_ML	HGNN-KNN	69.33 $\pm$ 0.2	68.89 $\pm$ 0.8	68.34 $\pm$ 0.6	68.11 $\pm$ 0.5	67.76 $\pm$ 1.0	67.40 $\pm$ 1.1	67.90 $\pm$ 0.7	66.12 $\pm$ 1.2
	HGNN- ϵ	69.13 $\pm$ 0.3	68.77 $\pm$ 0.8	68.19 $\pm$ 0.4	68.04 $\pm$ 0.8	67.42 $\pm$ 0.8	66.91 $\pm$ 1.0	67.64 $\pm$ 0.9	65.76 $\pm$ 1.3
Citeseer	HGNN-KNN	64.63 $\pm$ 0.2	63.34 $\pm$ 0.4	63.03 $\pm$ 0.6	62.90 $\pm$ 0.6	62.44 $\pm$ 1.6	61.68 $\pm$ 0.9	60.13 $\pm$ 0.7	58.05 $\pm$ 0.9
	HGNN- ϵ	62.12 $\pm$ 0.1	61.41 $\pm$ 0.5	60.90 $\pm$ 0.4	60.21 $\pm$ 0.4	59.13 $\pm$ 1.3	59.29 $\pm$ 0.7	59.34 $\pm$ 0.7	57.67 $\pm$ 1.5
Pubmed	HGNN-KNN	76.10 $\pm$ 0.4	75.55 $\pm$ 0.3	75.00 $\pm$ 0.6	74.48 $\pm$ 0.4	73.29 $\pm$ 0.9	72.23 $\pm$ 0.7	73.21 $\pm$ 0.5	71.59 $\pm$ 1.2
	HGNN- ϵ	75.54 $\pm$ 0.3	74.51 $\pm$ 0.4	74.04 $\pm$ 0.3	73.80 $\pm$ 0.8	72.32 $\pm$ 0.7	72.06 $\pm$ 0.4	72.93 $\pm$ 0.5	71.10 $\pm$ 1.0
DBLP-CA	HGNN-KNN	81.02 $\pm$ 0.3	80.62 $\pm$ 0.3	79.33 $\pm$ 0.5	78.81 $\pm$ 0.5	78.42 $\pm$ 0.6	77.56 $\pm$ 0.5	77.78 $\pm$ 0.6	76.29 $\pm$ 1.2
	HGNN- ϵ	80.25 $\pm$ 0.2	79.20 $\pm$ 0.3	78.88 $\pm$ 0.6	77.48 $\pm$ 0.7	77.17 $\pm$ 0.7	76.31 $\pm$ 0.6	77.12 $\pm$ 0.6	75.25 $\pm$ 1.4
NTU	HGNN-KNN	75.06 $\pm$ 0.2	73.99 $\pm$ 0.4	73.23 $\pm$ 0.5	72.92 $\pm$ 0.6	72.11 $\pm$ 1.0	72.43 $\pm$ 1.0	72.17 $\pm$ 0.5	71.22 $\pm$ 1.2
	HGNN- ϵ	73.73 $\pm$ 0.2	73.06 $\pm$ 0.7	72.61 $\pm$ 0.6	72.13 $\pm$ 0.5	70.30 $\pm$ 0.8	70.12 $\pm$ 1.2	70.49 $\pm$ 0.7	69.18 $\pm$ 1.4
ModelNet40	HGNN-KNN	89.91 $\pm$ 0.2	88.15 $\pm$ 0.4	87.93 $\pm$ 0.5	87.54 $\pm$ 0.7	86.67 $\pm$ 0.8	86.31 $\pm$ 0.7	86.97 $\pm$ 0.6	85.64 $\pm$ 0.9
	HGNN- ϵ	88.12 $\pm$ 0.3	87.49 $\pm$ 0.4	87.02 $\pm$ 0.4	86.46 $\pm$ 0.5	85.82 $\pm$ 0.9	85.20 $\pm$ 0.6	86.07 $\pm$ 0.6	84.19 $\pm$ 1.3

Table 4

Running time of various attacks in minutes. Algorithm runtime refers to the time from the start of the program's attack to the completion of the downstream task by the HGNNs, i.e., it includes training time and testing time.

Datasets	Model	Random	NDA	FGA-D	FGA	EpoAtk-E	MGHGA-D	MGHGA
Cora	HGNN-KNN	0.25	0.37	2.35	2.33	2.60	2.35	2.34
	HGNN- ϵ	0.25	0.37	2.35	2.33	2.58	2.35	2.34
Cora_ML	HGNN-KNN	0.25	0.35	3.33	3.31	3.56	3.34	3.33
	HGNN- ϵ	0.25	0.36	3.34	3.31	3.55	3.35	3.33
Citeseer	HGNN-KNN	0.27	0.40	4.15	4.10	4.86	4.19	4.15
	HGNN- ϵ	0.27	0.39	4.16	4.11	4.91	4.19	4.16
Pubmed	HGNN-KNN	1.60	3.54	14.31	13.98	15.23	14.48	14.08
	HGNN- ϵ	1.62	3.58	14.40	14.02	15.33	14.37	14.05
DBLP-CA	HGNN-KNN	2.31	4.25	19.58	19.20	21.58	19.93	19.35
	HGNN- ϵ	2.28	4.28	20.31	19.29	21.38	19.64	19.41
NTU	HGNN-KNN	0.25	0.44	2.56	2.50	3.01	2.56	2.55
	HGNN- ϵ	0.25	0.44	2.56	2.49	3.05	2.56	2.55
ModelNet40	HGNN-KNN	1.23	2.54	10.64	11.13	13.31	10.67	11.21
	HGNN- ϵ	1.26	2.58	10.66	11.16	13.20	10.71	11.18

model. The KNN method also reduces the negative impact of the attack in the perturbed hypergraph.

MGHGA improves the performance by 3% on average compared to Random. In particular, MGHGA improves the performance by 5% for Citeseer, which shows that MGHGA can add critically important perturbations with the same budget. HGNNs can better utilize global as well as longer-range contextual information when aggregating neighboring features, resulting in improved robustness of HGNNs over GNNs. Consequently, attacking HGNNs can be more difficult than attacking GNNs. MGHGA, as a preliminary exploration of the nontargeted poisoning attack on HGNNs, shows noteworthy performance compared to all other attacks.

Additionally, we investigate the effect of adding attack constraints (attacking the node with the largest node degree) on the attacks. When comparing MGHGA-D and MGHGA, we observe that the performance of MGHGA-D is not as good as that of MGHGA without constraints, although it can reduce the performance of HGNNs. For example, in Citeseer, the performance of MGHGA over MGHGA-D is increased for 2.08% and 2.67% in HGNN-KNN and HGNN- ϵ , respectively. The same observations hold for FGA-D and FGA. These results are somewhat expected, as intuitively, the unrestricted attack can maximize the efficiency of the attack.

5.5. Running time

In this section, we investigate the time performance and computational efficiency of several types of attacks, Table 4 shows the attack runtime and Table 5 shows the attack time complexity.

Specifically, Random time complexity is small $\mathcal{O}(d|V|)$, so the running time is short in each dataset. However, its performance is the worst of all attacks and is not usually considered in real attacks. Comparing FGA, EpoAtk-E and MGHGA, it can be found that the runtime of MGHGA is shorter than that of EpoAtk-E, which is similar to that of FGA, but enables more efficient attacks. For example, in Citeseer, the run times for FGA, EpoAtk-E and MGHGA are 4.10 min, 4.86 min and 4.15 min, respectively. As shown in Table 3, when the victim model is HGNN-KNN, their CSRs are 62.44%, 61.68% and 58.05% respectively. The time complexity of FGA and MGHGA shown in Table 5 is the same for both $\mathcal{O}(\epsilon\|H\| \cdot \|X\| + Td|V|)$. In Table 4, the running time of MGHGA is 0.01 to 0.09 min higher than that of FGA, and the results are in line with our expectations. Because FGA uses only single gradient information to attack the features, MGHGA uses momentum gradient to guide the attack to complete the modification of the features, which accumulates previous and current moment gradient information to help the attacker to cross the bad local optimums and thus generate better feature perturbations. The time complexity required for the MGHGA cumulative gradient operation is small and is not reflected in the time complexity mathematical expression. As the hypergraph feature size gets larger, MGHGA will have a slightly higher running time than FGA, but not significantly different. Since EpoAtk-E selects the optimal candidate solutions by retraining the HGNNs, it has the longest running time. The above results show that MGHGA can achieve efficient attacks with better or similar runtimes than other attacks.

Comparing the extended models FGA-D and MGHGA-D the results are also similar. The time complexity of FGA-D and MGHGA-D is $\mathcal{O}(\epsilon\|H\| \cdot \|X\| + Td|V| + |V|)$, which increases the time complexity of the sorted node degree over FGA and MGHGA by $\mathcal{O}(|V|)$.

Table 5

Time complexity of various attacks. Algorithm time complexity refers to the algorithm training time. In EpoAtk-E, HGNNs need to be trained to evaluate the candidate set, so the number of times HGNNs are trained is $t(r+1)$.

Model		Time complexity	
Random	NDA	$\mathcal{O}(d V)$	$\mathcal{O}(V + d V)$
FGA-D	FGA	$\mathcal{O}(t(H \cdot X + V + d V))$	$\mathcal{O}(t(H \cdot X + V + d V))$
EpoAtk-E		$\mathcal{O}(t(H \cdot X + V))$	
MGHGA-D	MGHGA	$\mathcal{O}(t(H \cdot X + V + d V))$	$\mathcal{O}(t(H \cdot X + V + d V))$

In most cases, we find that the runtime of the attack is positively correlated with the size of the nodes, and the runtime is longer when the node size is larger. For example, the numbers of nodes in descending order are DBLP-CA, Pubmed, ModelNet40, Citeseer, Cora_ML, Cora, and NTU, and the runtimes in descending order are ModelNet40, Citeseer, Cora_ML, NTU, and Cora. There is an exception where NTU > Cora. Intuitively, NTU is two-featured data, and HGNNs spend a longer time processing two-featured data than single-featured data [65].

5.6. Ablation experiment

5.6.1. Ablation study on different budgets

We further investigate the performance of various attacks under different budgets to verify the effectiveness of MGHGA. In this paper, budget represents the number of node features modified by the attacker, higher the budget higher is the number of node features manipulated by the attacker.

As shown in Fig. 4, we observe that MGHGA achieves satisfactory results with different budgets in all datasets. In Fig. 4(n), the CSR of FGA, EpoAtk-E and MGHGA are {87.64%, 85.82%, 85.29%}, {86.98%, 85.20%, 85.03%} and {86.62%, 84.19%, 84.14%} when the budget factor λ are {0.01, 0.05, 0.1}, respectively.

Furthermore, we investigated the effect of the budget on two constrained attacks (FGA-D and MGHGA-D). Fig. 4 shows that the increase in budget can negatively affect the constrained attacks. As an example, the CSR of FGA-D and MGHGA-D are {88.36%, 87.54%, 87.88%} and {88.12%, 86.97%, 87.40%} in Fig. 4(g), when λ is {0.01, 0.05, 0.1}. We postulate that the larger degree nodes (FGA-D and MGHGA-D attack the node with the larger node degree each time) have a limited impact on the attack performance. FGA-D and MGHGA-D attack features do not positively impact the attack when the budget is too large. As the budget increases, the CSR of restricted attacks decreases and then increases (lower CSR represents better performance), which is particularly evident for the Cora and ModelNet40 datasets.

5.6.2. Ablation study on K and ϵ

Fig. 5 shows the performance of our proposed attack for different parameters K . HGNN-KNN uses the K nearest neighbor method to select the neighbor nodes of each node. Specifically, for each node, it selects the nearest K neighbor nodes as its set of neighbors.

We observe that the performance of MGHGA is independent of the victim model HGNN-KNN parameter K , i.e., MGHGA still reduces the accuracy of HGNN-KNN regardless of K . For example, for NTU, MGHGA-D and MGHGA reduce the accuracy of {2.08%, 3.84%, 1.51%}, and {3.90%, 4.27%, 3.81%}, respectively, when K is {5, 10, 15}. Additionally, the CSR of attacks is positively related to the accuracy of HGNNs in clean hypergraphs for most cases. In other words, if HGNNs have high accuracy in clean hypergraph classification, then HGNNs also have high accuracy in perturbed hypergraph classification.

Next, we investigate the effect of parameter ϵ on our attack, and the results are shown in Fig. 6. HGNN- ϵ uses the ϵ -neighbor method to select the neighbor nodes of each node. Specifically, for each node, it selects all neighboring nodes with a distance within ϵ as its set of neighbors.

For each dataset, MGHGA can achieve reduced HGNN- ϵ CSR. Specifically, the average performance of MGHGA-D and MGHGA in Citeseer is 3.03% and 4.43% under all ϵ , respectively. The above results indicate

that the victim model parameters do not affect our attack. Previous work has shown that the parameters K and ϵ do not affect the performance of HGNNs on classification tasks, i.e., HGNNs have good stability [6]. We hypothesize that the parameters K and ϵ do not affect our attack due to the stability of HGNNs.

5.6.3. Ablation study on different decay factors

The decay factor μ is an important parameter of our attack, as depicted in Fig. 7. Note that the MGHGA degrades to FGA when $\mu = 0$. We observe that the attack performance first increases and reaches a plateau or decreases as μ increases. For example, in Fig. 7(a) with NTU-1, the CSR of MGHGA is {72.11%, 71.21%, 71.23%} when μ is {0, 1, 1.4}, respectively. In addition, our attack performs best when $\mu = 1$ for discrete and continuous feature datasets in most cases. Intuitively, when μ is small, the momentum gradient mainly depends on the gradient of the HGNNs at the t moment, and the performance of MGHGA is similar to that of the normal gradient attack FGA. As μ increases, the gradient of the MGHGA depends on the gradients of the previous moments and the current moment, and this combining of the gradients starts to improve the performance of the MGHGA. However, as μ continues to increase, the momentum gradient relies heavily on the gradient of the previous moments and ignores the feedback from the gradient of the latest moment, and MGHGA performance degrades. The above results indicate that we can improve the attack performance by using the momentum gradient.

5.6.4. Transferability of MGHGA

Many previous works have assumed that the victim model and the surrogate model maintain the same knowledge and deep learning model, hence the attack lacks realism. In order to investigate the effects of victim modeling and surrogate modeling on the effectiveness of MGHGA attacks, we verify the transferability of our attack.

The confusion matrix (Fig. 8) illustrates the performance of MGHGA for various surrogate and victim models. Note that the various combinations of surrogate and victim models do not affect the MGHGA performance. Specifically, for Citeseer, the performance of MGHGA improved by 0.16% when the surrogate model and victim models are HGNN-KNN and HGNN- ϵ , respectively. We believe that MGHGA completes the attack before the victim HGNNs are trained and do not need to access the training parameters of the victim model, so the choice of surrogate and victim models does not affect the performance of MGHGA. In addition, there are differences in the way the hypergraphs of HGNN-KNN and HGNN- ϵ are modeled, making their accuracies in the classification task different, and the MGHGA performance differs under different combinations of HGNNs.

5.6.5. Case study - MGHGA performance under other metrics

This section conducts a case study to validate the effectiveness of MGHGA using MAE and Logloss metrics in Cora and NTU datasets. Where, the performance is verified in Cora and NTU in node classification and visual object classification tasks, respectively. The results are shown in Table 6.

In Cora and NTU, MGHGA achieves the highest attack scores in both MAE and Logloss metrics, indicating that its attack performance is significantly better than other baseline methods. For example, in the NTU dataset (HGNN- ϵ), MGHGA's MAE (0.17) and Logloss (1.49) are

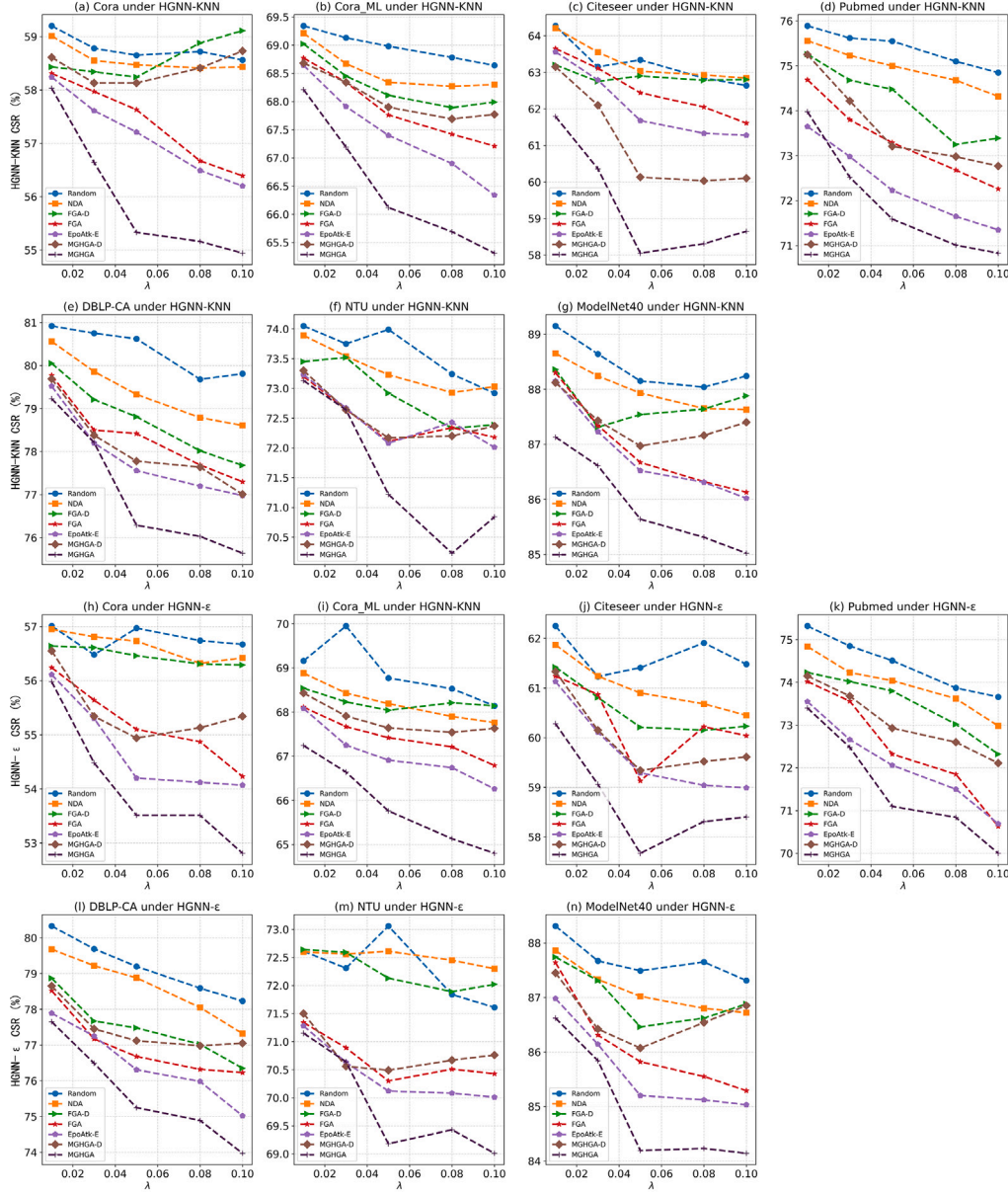


Fig. 4. The attack performance under different budget factors λ . The larger the λ the larger the number of node features modified by the attack.

Table 6

Performance of several types of attacks under MAE and LogLoss. For MAE and LogLoss, the higher the attacker score, the better the attack performance. For all cases, the best results are given in bold style.

Datasets	HGNN-KNNs	Metric	Random	NDA	FGA-D	FGA	EpoAtk-E	MGHGA-D	MGHGA
Cora	HGNN-KNN	MAE	0.12	0.14	0.22	0.26	0.27	0.25	0.30
		Logloss	1.34	1.65	2.02	2.12	2.19	2.10	2.26
	HGNN- ϵ	MAE	0.11	0.16	0.23	0.26	0.26	0.24	0.29
		Logloss	1.35	1.69	2.06	2.13	2.15	2.08	2.18
NTU	HGNN-KNN	MAE	0.09	0.11	0.13	0.15	0.16	0.14	0.18
		Logloss	0.91	1.07	1.35	1.40	1.49	1.40	1.56
	HGNN- ϵ	MAE	0.08	0.11	0.14	0.15	0.15	0.14	0.17
		Logloss	0.89	1.11	1.30	1.33	1.42	1.32	1.49

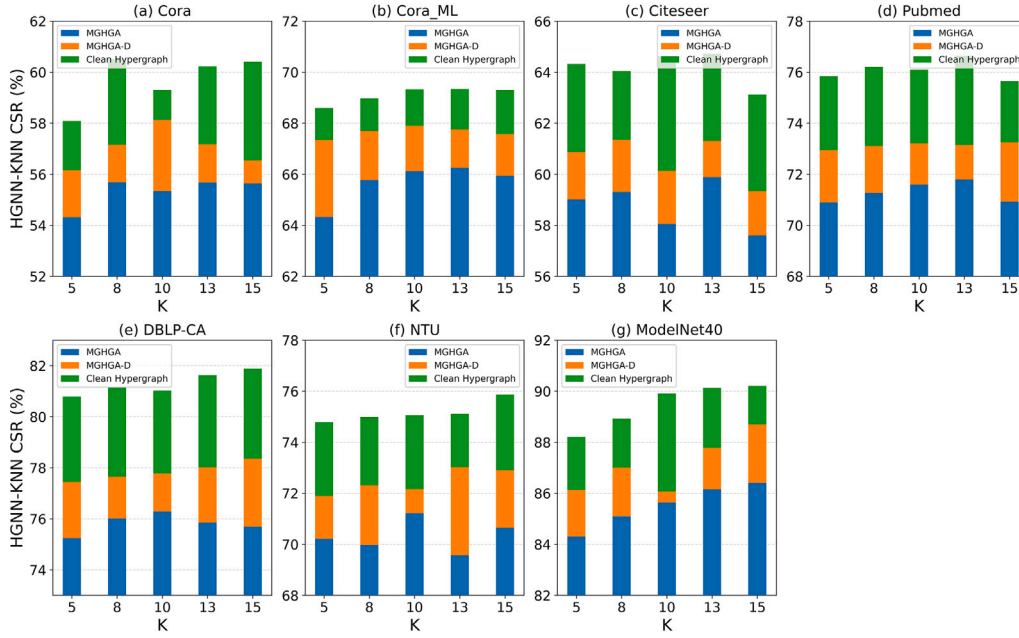


Fig. 5. The attack performance for different parameters K . The parameter K denotes the number of nodes connected to the same hyperedge.

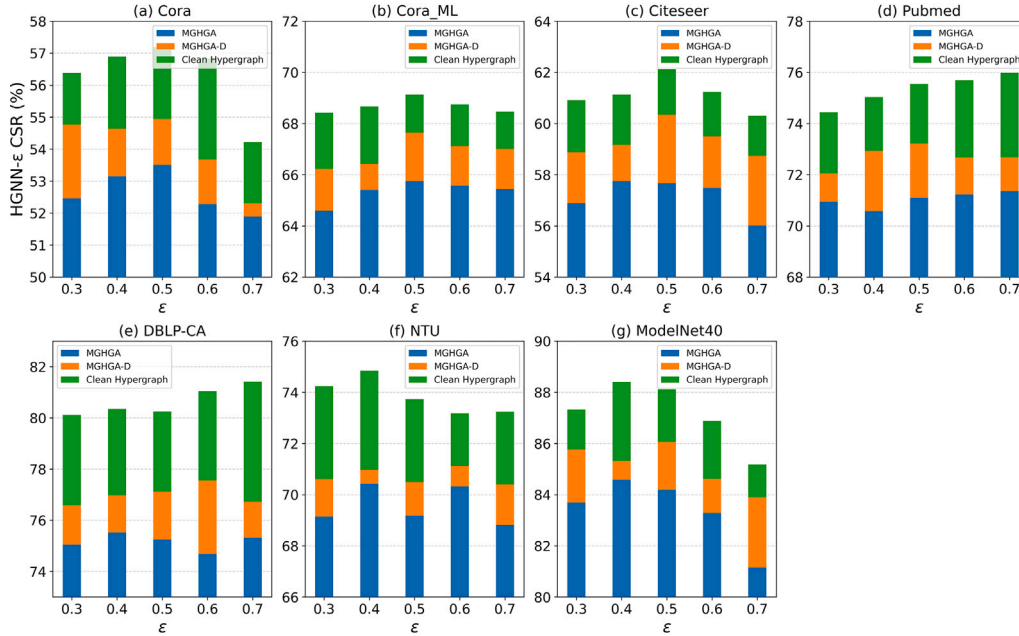


Fig. 6. The attack performance for different parameters ϵ . The larger the ϵ , the larger the number of hyperedge connected nodes.

higher than all the compared methods, and it is especially advantageous to graph-based methods (e.g., FGA, NDA).

The above results indicate that compared with traditional graph attack methods (e.g., FGA), MGHGA more efficiently destroys the higher-order correlation properties of the hypergraph by adaptively adjusting the hyperedge perturbation strategy through the momentum gradient.

6. Conclusions and future work

Our work shows that HGNNs are vulnerable to untargeted poisoning attacks. This paper presents the first untargeted attack on HGNNs, denoted MGHGA. Considering the training differences between HGNN and GNNs, MGHGA uses surrogate models to modify node features before hypergraph modeling. Specifically, MGHGA uses the momentum

gradient mechanism to select the features of the attack nodes. MGHGA uses different methods to update discrete and continuous features in the feature generation module. Extensive experimental results show that MGHGA can achieve good attack performance in node and visual object classification tasks.

In this paper, we only discuss the vulnerability of HGNNs. However, MGHGA has drawbacks. For example, MGHGA is a white-box attack that accesses the HGNNs' training parameters during the attack process. Sometimes, the attacker can only access specific or none of the parameters, leading to the failure of MGHGA. In future work, we will consider two main aspects: (1) investigating the robustness of HGNNs against gray-box and black-box attacks, and (2) improving the HGNNs' robustness against untargeted attacks. We hope that MGHGA is the first

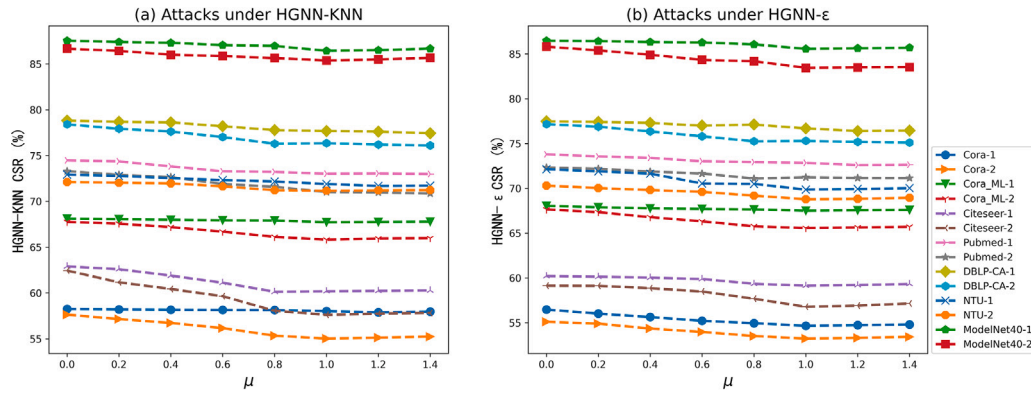


Fig. 7. The attack performance for different decay factors μ . The μ represents the momentum gradient depends mainly on whether the gradient of the HGNNs is at the current moment or at a previous moment. To facilitate the presentation of the results, '1' and '2' in the legend denote the results under MGHGA-D and MGHGA, respectively. For example, cora-1 denotes the results of MGHGA-D under the cora dataset.

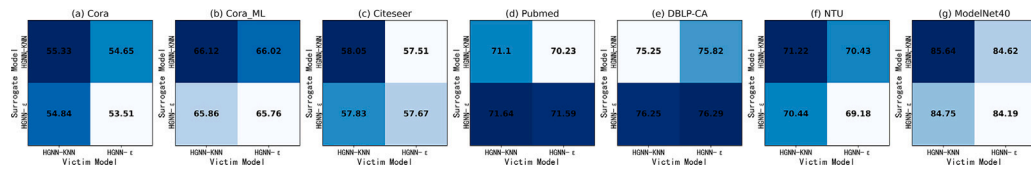


Fig. 8. The transferability of MGHGA. The x-axis represents the victim model, and the y-axis represents the surrogate model.

step in opening up novel research avenues for studying HGNN attacks and defenses.

CRediT authorship contribution statement

Yang Chen: Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Stjepan Picek:** Writing – review & editing, Supervision, Investigation. **Zhonglin Ye:** Visualization, Investigation. **Zhaoyang Wang:** Investigation. **Haixing Zhao:** Validation, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2020YFC1523300), Construction Project for Innovation Platform of Qinghai Province (No. 2022ZJT02) and Project of Scientific Research Initiation Fund of Shandong Technology and Business University (No. BS202454). In addition, thanks to Stjepan Picek for his contribution to this manuscript.

Data availability

The processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study. In future, the processed data that support the findings of this study are available from the corresponding author upon reasonable request.

References

- [1] Tianjun Yao, Jiaqi Sun, Defu Cao, Kun Zhang, Guangyi Chen, MuGSI: Distilling GNNs with multi-granularity structural information for graph classification, in: *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 709–720.
- [2] Zhaowei Liu, Dong Yang, Yingjie Wang, Mingjie Lu, Ranran Li, EGNN: Graph structure learning based on evolutionary computation helps more in graph neural networks, *Appl. Soft Comput.* 135 (2023) 110040.
- [3] Sitao Luan, Chenqing Hua, Minkai Xu, Qincheng Lu, Jiaqi Zhu, Xiao-Wen Chang, Jie Fu, Jure Leskovec, Doina Precup, When do graph neural networks help with node classification? Investigating the homophily principle on node distinguishability, *Adv. Neural Inf. Process. Syst.* 36 (2024).
- [4] Xiaohu Li, Zide Fan, Feilong Huang, Xuming Hu, Yawen Deng, Lei Wang, Xinyu Zhao, Graph neural network with curriculum learning for imbalanced node classification, *Neurocomputing* 574 (2024) 127229.
- [5] Kartik Sharma, Yeon-Chang Lee, Sivagami Nambi, Aditya Salian, Shlok Shah, Sang-Wook Kim, Srikanth Kumar, A survey of graph neural networks for social recommender systems, *ACM Comput. Surv.* 56 (10) (2024) 1–34.
- [6] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, Yue Gao, Hypergraph neural networks, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3558–3565.
- [7] Haojie Sun, Zhaowei Liu, Shengqiang Wang, Haiyang Wang, Adaptive attention-based graph representation learning to detect phishing accounts on the ethereum blockchain, *IEEE Trans. Netw. Sci. Eng.* 11 (3) (2024) 2963–2975.
- [8] Yi Han, Bin Zhou, Jian Pei, Yan Jia, Understanding importance of collaborations in co-authorship networks: A supportiveness analysis approach, in: *Proceedings of the SIAM International Conference on Data Mining*, 2009, pp. 1111–1122.
- [9] Junwu Chen, Philippe Schwallier, Molecular hypergraph neural networks, *J. Chem. Phys.* 160 (14) (2024).
- [10] Sunwoo Kim, Soo Yong Lee, Yue Gao, Alessia Antelmi, Mirko Polato, Kijung Shin, A survey on hypergraph neural networks: an in-depth and step-by-step guide, in: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 6534–6544.
- [11] Jinyin Chen, Ximin Zhang, Haibin Zheng, Backdoor attack on dynamic link prediction, in: *Attacks, Defenses and Testing for Deep Learning*, Springer, 2024, pp. 119–142.
- [12] Ryo Meguro, Hiroya Kato, Shintaro Narisada, Seira Hidano, Kazuhide Fukushima, Takuo Suganuma, Masahiro Hiji, Gradient-based clean label backdoor attack to graph neural networks, in: *ICISSP*, 2024, pp. 510–521.
- [13] Mauro Conti, Jiaxin Li, Stjepan Picek, Jing Xu, Label-only membership inference attack against node-level graph neural networks, in: *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security, AISec'22*, Association for Computing Machinery, New York, NY, USA, 2022, pp. 1–12.
- [14] He Zhao, Zhiwei Zeng, Yongwei Wang, Deheng Ye, Chunyan Miao, HGAttack: Transferable heterogeneous graph adversarial attack, 2024, arXiv preprint arXiv: 2401.09945.
- [15] Jiazhu Dai, Weifeng Zhu, Xiangfeng Luo, A targeted universal attack on graph convolutional network by using fake nodes, *Neural Process. Lett.* 54 (4) (2022) 3321–3337.

- [16] Mohammed Aburidi, Roummel Marcia, Topological adversarial attacks on graph neural networks via projected meta learning, in: 2024 IEEE International Conference on Evolving and Adaptive Intelligent Systems, EAIS, IEEE, 2024, pp. 1–8.
- [17] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, Jiliang Tang, Graph Structure Learning for Robust Graph Neural Networks, 2020.
- [18] Yang Chen, Zhonglin Ye, Zhaoyang Wang, Haixing Zhao, Imperceptible graph injection attack on graph neural networks, *Complex Intell. Syst.* 10 (1) (2024) 869–883.
- [19] Chao Hu, Ruishi Yu, Binqi Zeng, Yu Zhan, Ying Fu, Quan Zhang, Rongkai Liu, Heyuan Shi, HyperAttack: Multi-gradient-guided white-box adversarial structure attack of hypergraph neural networks, 2023.
- [20] Junyuan Fang, Haixian Wen, Jiajing Wu, Qi Xuan, Zibin Zheng, K Tse Chi, Gani: Global attacks on graph neural networks via imperceptible node injections, *IEEE Trans. Comput. Soc. Syst.* (2024).
- [21] Kai Zhao, Qiyu Kang, Yang Song, Rui She, Sijie Wang, Wee Peng Tay, Adversarial robustness in graph neural networks: A Hamiltonian approach, *Adv. Neural Inf. Process. Syst.* 36 (2024).
- [22] Wei Kong, Yufang Xu, Shuaiqun Wang, Kai Wei, Gen Wen, Yaling Yu, Yuemin Zhu, A novel longitudinal phenotype-genotype association study based on deep feature extraction and hypergraph models for Alzheimer's disease, *Biomolecules* 13 (5) (2023).
- [23] Yue Gao, Zizhao Zhang, Haojie Lin, Xibin Zhao, Shaoyi Du, Changqing Zou, Hypergraph learning: Methods and practices, *IEEE Trans. Pattern Anal. Mach. Intell.* 44 (5) (2022) 2548–2566.
- [24] Maximilian T. Fischer, Devanshu Arya, Dirk Streeb, Daniel Seebacher, Daniel A. Keim, Marcel Worring, Visual analytics for temporal hypergraph model exploration, *IEEE Trans. Vis. Comput. Graphics* 27 (2) (2021) 550–560.
- [25] Zhongtian Ma, Zhiguo Jiang, Haopeng Zhang, Hyperspectral image classification using feature fusion hypergraph convolution neural network, *IEEE Trans. Geosci. Remote Sens.* 60 (2022).
- [26] Thomas N. Kipf, Max Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.
- [27] Sajjad Heydari, Lorenzo Livi, Message passing neural networks for hypergraphs, in: *Artificial Neural Networks and Machine Learning*, vol. 13530, 2022, pp. 583–592.
- [28] Yuchi Huang, Qingshan Liu, Dimitris Metaxas, Video object segmentation by hypergraph cut, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 1738–1745.
- [29] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, Le Song, Adversarial attack on graph structured data, in: *International Conference on Machine Learning*, PMLR, 2018, pp. 1115–1124.
- [30] Hussain Hussain, Meng Cao, Sandipan Sikdar, Denis Helic, Elisabeth Lex, Markus Strohmaier, Roman Kern, Adversarial inter-group link injection degrades the fairness of graph neural networks, 2022, arXiv preprint arXiv:2209.05957.
- [31] Jing Huang, Xiaolin Huang, Jie Yang, Residual enhanced multi-hypergraph neural network, in: 2021 IEEE International Conference on Image Processing, IEEE, 2021, pp. 3657–3661.
- [32] Yixian Wang, Zhaowei Liu, Jindong Xu, Weiqing Yan, Heterogeneous network representation learning approach for ethereum identity identification, *IEEE Trans. Comput. Soc. Syst.* 10 (3) (2023) 890–899.
- [33] Ali Behrouz, Farnoosh Hashemi, Sadaf Sadeghian, Margo Seltzer, CAT-walk: Inductive hypergraph learning via set walks, *Adv. Neural Inf. Process. Syst.* 36 (2024).
- [34] Mikhail Hayhoe, Hans Matthew Riess, Michael M Zavlanos, Victor Preciado, Alejandro Ribeiro, Transferable hypergraph neural networks via spectral similarity, in: *Learning on Graphs Conference*, PMLR, 2024, pp. 1–18.
- [35] Meng Wang, Xueliang Liu, Xindong Wu, Visual classification by 11 hypergraph modeling, *IEEE Trans. Knowl. Data Eng.* 27 (9) (2015) 2564–2574.
- [36] Sheng Huang, Mohamed Elhoseiny, Ahmed Elgammal, Dan Yang, Learning hypergraph-regularized attribute predictors, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 409–417.
- [37] Quan Fang, Jitao Sang, Changsheng Xu, Yong Rui, Topic-sensitive influencer mining in interest-based social media networks via hypergraph learning, *IEEE Trans. Multimed.* 16 (3) (2014) 796–812.
- [38] Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, Yue Gao, Dynamic hypergraph neural networks, in: 2019 International Joint Conferences on Artificial Intelligence, 2019, pp. 2635–2641.
- [39] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, Partha Talukdar, Hypergc: A new method for training graph convolutional networks on hypergraphs, *Adv. Neural Inf. Process. Syst.* 32 (2019).
- [40] Dengfeng Liu, Zhiqiang Pan, Shengze Hu, Fei Cai, Distance enhanced hypergraph learning for dynamic node classification, *Neural Process. Lett.* 56 (5) (2024) 231.
- [41] Yumeng Song, Yu Gu, Tianyi Li, Jianzhong Qi, Zhenghao Liu, Christian S. Jensen, Ge Yu, CHGNN: A semi-supervised contrastive hypergraph learning network, *IEEE Trans. Knowl. Data Eng.* 36 (9) (2024) 4515–4530.
- [42] Jinyin Chen, Yangyang Wu, Xuanheng Xu, Yixian Chen, Haibin Zheng, Qi Xuan, Fast gradient attack on network embedding, 2018, arXiv preprint arXiv:1809.02797.
- [43] Junyuan Fang, Haixian Wen, Jiajing Wu, Qi Xuan, Zibin Zheng, Chi K Tse, GANI: Global attacks on graph neural networks via imperceptible node injections, 2022, arXiv preprint arXiv:2210.12598.
- [44] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, Liming Zhu, Adversarial examples on graph data: Deep insights into attack and defense, 2019, arXiv preprint arXiv:1903.01610.
- [45] Daniel Zügner, Stephan Günnemann, Adversarial attacks on graph neural networks via meta learning, 2019, arXiv preprint arXiv:1902.08412.
- [46] Daniel Zügner, Amir Akbarnejad, Stephan Günnemann, Adversarial attacks on neural networks for graph data, in: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2847–2856.
- [47] Heyuan Shi, Binqi Zeng, Ruishi Yu, Yulin Yang, Zijian Zouxia, Chao Hu, Ronghua Shi, H3NI: Non-target-specific node injection attacks on hypergraph neural networks via genetic algorithm, *Neurocomputing* 613 (2025) 128746.
- [48] Kaiyang Wang, Huaxin Deng, Yijia Xu, Zhonglin Liu, Yong Fang, Multi-target label backdoor attacks on graph neural networks, *Pattern Recognit.* 152 (2024) 110449.
- [49] Feilong Cao, Qiyang Chen, Hailiang Ye, A new strategy of graph structure attack: Multi-view perturbation candidate edge learning, *IEEE Trans. Netw. Sci. Eng.* 11 (5) (2024) 4158–4168.
- [50] Mingxuan Ju, Yujie Fan, Chuxu Zhang, Yanfang Ye, Let graph be the go board: Gradient-free node injection attack for graph neural networks via reinforcement learning, 2022, arXiv preprint arXiv:2211.10782.
- [51] Ansh Kumar Sharma, Rahul Kukreja, Mayank Kharbanda, Tanmoy Chakraborty, Node injection for class-specific network poisoning, 2023, arXiv preprint arXiv:2301.12277.
- [52] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, Tom Goldstein, Poison frogs! targeted clean-label poisoning attacks on neural networks, *Adv. Neural Inf. Process. Syst.* 31 (2018).
- [53] Binghui Wang, Tianxiang Zhou, Minhua Lin, Pan Zhou, Ang Li, Meng Pang, Cai Fu, Hai Li, Yiran Chen, Evasion attacks to graph neural networks via influence function, 2020, arXiv preprint arXiv:2009.00203.
- [54] He Zhang, Xingliang Yuan, Chuan Zhou, Shirui Pan, Projective ranking-based gnn evasion attacks, *IEEE Trans. Knowl. Data Eng.* (2022).
- [55] Chao Jiang, Yi He, Richard Chapman, Hongyi Wu, Camouflaged poisoning attack on graph neural networks, in: *Proceedings of the 2022 International Conference on Multimedia Retrieval*, New York, NY, USA, 2022, pp. 451–461.
- [56] Houxiang Fan, Binghui Wang, Pan Zhou, Ang Li, Zichuan Xu, Cai Fu, Hai Li, Yiran Chen, Reinforcement learning-based black-box evasion attacks to link prediction in dynamic graphs, in: 2021 IEEE 23rd Int Conf on High Performance Computing & Communications, IEEE, 2021, pp. 933–940.
- [57] Jinyin Chen, Yixian Chen, Haibin Zheng, Shijing Shen, Shaoqing Yu, Dan Zhang, Qi Xuan, Mga: momentum gradient attack on network, *IEEE Trans. Comput. Soc. Syst.* 8 (1) (2020) 99–109.
- [58] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy, Explaining and harnessing adversarial examples, 2014, arXiv preprint arXiv:1412.6572.
- [59] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, Tina Eliassi-Rad, Collective classification in network data, *AI Mag.* 29 (3) (2008) 93.
- [60] Eli Chien, Chao Pan, Jianhao Peng, Olga Milenkovic, You are AllSet: A multiset function framework for hypergraph neural networks, in: *International Conference on Learning Representations*, 2022.
- [61] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, Ming Ouhyoung, On visual similarity based 3D model retrieval, in: *Computer Graphics Forum*, vol. 22, no. 3, Wiley Online Library, 2003, pp. 223–232.
- [62] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, Jianxiong Xiao, 3D shapenets: A deep representation for volumetric shapes, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.
- [63] Xiaoyun Wang, Minhao Cheng, Joe Eaton, Cho-Jui Hsieh, S Felix Wu, Fake node attacks on graph convolutional networks, *J. Comput. Cogn. Eng.* 1 (4) (2022) 165–173.
- [64] Xixun Lin, Chuan Zhou, Jia Wu, Hong Yang, Haibo Wang, Yanan Cao, Bin Wang, Exploratory adversarial attacks on graph neural networks for semi-supervised node classification, *Pattern Recognit.* 133 (2023) 109042.
- [65] Yue Gao, Yifan Feng, Shuyi Ji, Rongrong Ji, HGNN+: General hypergraph neural networks, *IEEE Trans. Pattern Anal. Mach. Intell.* 45 (3) (2023) 3181–3199.



Yang Chen received his Ph.D. degree in Computer Science and Technology at Qinghai Normal University. He is currently a lecturer with the School of Computer Science and Technology, Shandong Technology and Business University. His current research interests include the adversarial attack and defense based on deep learning.



Stjepan Picek received the Ph.D. degrees in computer science from Radboud University, Nijmegen, The Netherlands, and University of Zagreb, Zagreb, Croatia, in 2015. He was with the COSIC Group, KU Leuven, Leuven, Belgium. He was a Postdoctoral Researcher with the MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA. He is currently an associate professor in the Digital Security group. He studies the intersection of cybersecurity and artificial intelligence.



Zhaoyang Wang received her B.S. degree in Mathematics and Applied Mathematics and her M.S. degree in Theoretical Computer Science from Qinghai Normal University. She is currently pursuing his Ph.D. degree in Computer Science and Technology at Qinghai Normal University. Her main research interests include complex network, high-order network theory and application.



Zhonglin Ye received the Ph.D. degree from the Shaanxi Normal University, Xian, China, in 2019. He is currently a postdoctoral fellow at the University of Regina, Canada. His main research interests include data mining, knowledge discovery, link prediction, network representation, and machine learning.



Haixing Zhao received the Ph.D. degree of the Computer Software and Theory from the Northwestern Polytechnical University, Xian, China, in 2004, and the Ph.D. degree of the Applied Mathematics from Universiteit Twente, Enschede, Holland, in 2005. He is currently a Professor with the School of Computer Science, Qinghai Normal University. His main research interests include complex networks, semantic networks and machine translation, hypergraph theory and database, and network reliability.