Projective Ranking-Based GNN Evasion Attacks

He Zhang†, Xingliang Yuan†, Chuan Zhou†, and Shirui Pan†

Abstract—Graph neural networks (GNNs) offer promising learning methods for graph-related tasks. However, GNNs are at risk of adversarial attacks. Two primary limitations of the current evasion attack methods are highlighted: (1) The current GradArgmax ignores the “long-term” benefit of the perturbation. It is faced with zero-gradient and invalid benefit estimates in certain situations. (2) In reinforcement learning-based attack methods, the learned attack strategies might not be transferable when the attack budget changes. To this end, we first formulate the perturbation space and propose an evaluation framework and the projective ranking method. We aim to learn a powerful attack strategy then adapt it as little as possible to generate adversarial samples under dynamic budget settings. In our method, based on mutual information, we rank and assess the attack benefits of each perturbation for an effective attack strategy. By projecting the strategy, our method dramatically minimizes the cost of learning a new attack strategy when the attack budget changes. In the comparative assessment with GradArgmax and RL-S2V, the results show our method owns high performance and effective transferability. The visualization of our method also reveals various attack patterns in the generation of adversarial samples.

Index Terms—Adversarial attacks, graph neural networks, graph classification

1 INTRODUCTION

Graphs consist of nodes and edges defined between these nodes. As an abstract data type, graphs have powerful modelling capabilities. By extracting attributes from entities and describing their relationships, graphs can represent a range of objects or technical systems in real-world scenarios, such as drug molecules [1], biological networks [2], and traffic networks [3], [4]. Graph neural networks (GNNs) have proven to be effective graph learning methods [5] for exploring graph data and have demonstrated promising performance in node classification [6], [7], link prediction [8], anomaly detection [9], [10], and graph classification [11]. For example, in biological chemistry, the GNNs are engaged in recognizing the chemical properties of molecules.

In practice, GNNs raise urgent security concerns, although they have garnered considerable attention in the context of complex graph-structured data applications [12], [13], [14], [15]. Generally, graph neural networks are regarded as the generalization of deep neural networks into graph data. However, GNNs inherit the vulnerability of deep neural networks despite possessing adequate expressive power. Several recent studies demonstrate that adversaries can attack GNNs [16], [17], [18], [19], [20], [21]. Among others, evasion attacks [22], [23] are notoriously dangerous. Attackers could perturb test samples to generate adversarial samples of a victim model trained on original clean data. On adversarial samples, the victim model will give incorrect results, different from those of clean samples.

Prior work on GNN evasion attacks has demonstrated that well-designed slight perturbations are able to significantly degrade the performance of victim models [22]. This type of attack is tricky since adversarial samples are similar to clean samples in appearance [17]. Adding a few elaborate adversarial edges can dramatically reduce the classification accuracy of the victim models [23]. To create an adversarial example, attackers can change the features of few nodes or add/remove a few edges in a clean graph. A budget $k$ with positive integer values controls the number of these perturbation operations to make their perturbations stealthy. Moreover, once the models are deployed, it is practical for attackers to launch evasion attacks at any time [19].

The existing studies of evasion attacks on GNN models mainly focus on node classification and link prediction, while few of them are designed for graph classification [16], [17], [19]. We note there are some limitations in those methods for graph classification, which need to be solved to improve the attack practicality.

- In the vanilla gradient-based attack method GradArgmax [22], the greedy mechanism uses the gradient on the graph in the perturbation process to make the perturbation decision. Although the gradient information could approximate the attack benefit to some extent, it ignores the long-term benefit of operation at each step [22]. Moreover, GradArgmax is not efficient
since it needs the up-to-date gradient information to complete each step of all the perturbation operations.

- **Attack methods** [22], [24] use reinforcement learning (RL) to perturb clean samples and use the final predicted result on adversarial samples to estimate the attack benefit. When the perturbation budget changes, they have to be re-trained to learn an attack strategy suitable for the new budget. In addition, we observe that RL-S2V [22], a known attack method, does not appear to be able to generate adversarial samples at high attack success rates when the perturbation budget increases [24].

Motivated by these observations, we summarize the challenges in building an effective evasion attack against GNNs for graph classification: (1) in the generation of powerful adversarial samples, how to measure the attack benefits of each operation based on the clean graph, and (2) how to adapt the learned attack strategy to changed perturbation settings (e.g., budget) with least effort.

To address these challenges, we first formulate the perturbation space composed of all possible perturbation operations and present four properties of it. Then, we propose an evaluation framework consisting of three principles decomposed from the adversary’s expectation. We use these three principles as the design goal of our method and employ them to evaluate typical evasion attack methods. In our evaluation framework, we consider a basic fact in the evasion attack: an intelligent attacker always expects to choose the perturbation operation which will bring the most significant attack benefit at each perturbation step. This fact includes three critical principles for evasion attack: attack benefit, operation ranking and baseline graph. Finally, we propose the projective ranking method as an instantiation under the evaluation framework.

In our method, we expect to find a suitable metric that considers the attack benefits of each operation to rank them and then use this ranking directly to generate adversarial samples under different perturbation budget settings. Towards the first challenge, we employ the mutual information (MI) between elements of perturbation space and attackers’ goals as a measure to evaluate the importance of each possible operation. We regard the perturbation space ranking as the learned attack strategy. Inspired by the projected gradient descent method, we propose to project the learned attack strategy into practical attackers’ perturbations with the specified budget for the second challenge. Through this projection operation, the adjustment cost of the learned attack strategy is almost zero.

Our contributions are summarized below:

- **We conduct evaluations on several real world datasets and a synthesized dataset, and our attack method achieves high attack performance and effective transferability. The visualization of adversarial samples present various attack patterns and reveals the vulnerability of the victim model.**

In this article, we review different attack methods for graph classification in Section 2 and introduce the graph classification and evasion attacks in Section 3. In Section 4, we first formulate the perturbation space and show its four properties. Then we propose the evaluation framework of evasion attack methods and show the instantiation design of our method. Next, we show the details of the projective ranking method. In Section 5, we use the evaluation framework to make a comparison evaluation of typical methods. In Section 6, we experimentally demonstrate the high attack performance and effective transferability of our method, and the attack patterns in our adversarial samples reveal the weaknesses of the victim model. Finally, we show the conclusion and discuss the future works in Section 7.

## 2 RELATED WORK

Current research shows that graph neural networks are vulnerable to adversarial attacks. According to the intention of attackers, the adversarial attack methods for graph classification include evasion attacks, backdoor attacks, and poisoning attacks.

### 2.1 Evasion Attack Methods on GNNs

Evasion attacks perturb the inputs of GNNs during their inference phase and intend to degrade the performance of the victim model. Hence, the more the classification accuracy of the victim model decreases, the better the performance of an attack method. As the most intuitive approach, RandomSampling randomly perturbs the graph structure or nodes’ features of clean graphs to generate adversarial samples. However, the attack performance of this method is lower than adding well-designed perturbations. In the GradArgmax method, the attackers own the ability to access the victim model. They first calculate the gradient information of the loss function on each possible perturbation operation, then use the gradient information to choose the perturbation operation that maximizes the loss of the victim model.

To better describe the perturbations on graph data, Dai et al. [22] proposed to employ the Markov decision process to model the whole perturbation process and use reinforcement learning to obtain an attack strategy on current clean samples. The graph data in the perturbation process is modeled as the state of reinforcement learning. The action space comprises all perturbation operations in the current state, and the final attack result defines the reward function. Benefiting from the modelling and learning ability of reinforcement learning, the attack performance of RL-S2V outperforms that of RandSampling and GradArgmax on synthetic data [22].

To improve the stealthiness of attack, Ma et al. proposed ReWatt to redefine the action space of reinforcement learning [24]. The perturbations generated by ReWatt are more unnoticeable since the edge numbers of adversarial samples are equal to those of clean samples. Besides the above attack methods, attackers could also attack the components of...
TABLE 1
Evasion Attack Methods for Graph Classification

| Method       | Decision-making information | Operations          | Attack Goal          | Transferability of Attack |
|--------------|-----------------------------|---------------------|----------------------|---------------------------|
| Rand*        | Random                      | Add/Remove          | Untargeted attack    | Unseen samples            |
| GradArgmax   | Gradient of loss function   | Add/Remove          | Untargeted/Targeted  | -                         |
| RL-S2V       | Reward in RL                | Add                 | Untargeted attack    | Unseen samples            |
| Rewatt       | Reward in RL                | Rewrite             | Untargeted/Targeted  | -                         |
| HGP*         | Gradient of selection function | Add/Remove modify | Untargeted attack    | Unseen samples            |
| Ours         | Mutual information          | Add                 | Untargeted/Targeted  | -                         |

*Rand means the RandomSampling. HGP means the method for attacking the pooling operation in Hierarchical Graph Pooling Neural Networks.

GNNs if they know that a specified operation is included in the victim model. For example, some methods introduce hierarchical pooling operations in the models, which select typical nodes to compress the node number and determine the structure of the coarse graph [25], [26], [27].

Tang et al. proposed to attack the selection operation in pooling operation [28]. By training a surrogate model with hierarchical pooling on some clean data, the attackers obtain the selecting function that simulates the victim model’s hierarchical pooling. Based on this selection function, the attackers could obtain the adversarial samples that invalidate the pooling in the victim model. However, this method may not be available for attacking models that employ global pooling operations. A comparison of representative evasion attack methods is summarized in Table 1.

2.2 Differences in Evasion Attacks

In addition to the differences in technical methods, the evasion attack methods also have differences in perturbation space, the definition of imperceptible perturbations, the attacker’s knowledge, and the attack goal. In the aspect of perturbation space, the attackers usually make perturbations by adding fake nodes with the fake features [29], modifying node features [30], adding or deleting edges [31]. In graph classification evasion attacks, ReWatt, RL-S2V, RandSampling and GradArgmax only modify graph structure. The attack method in [28] makes perturbations in node features or graph structure.

In the aspect of imperceptible perturbations, the distance between the original clean graph and the modified adversarial graph is usually limited by the perturbation budget, which is defined as the number of nodes modified or distance of feature vectors, or the number of edges modified by adding/deleting/rewriting. In the aspect of attackers’ knowledge, based on how much information an attacker knows about the victim model, the attack methods are generally divided into white-box, grey-box, and black-box attacks. A white-box attack means the attackers can access all information about the victim model like architecture, parameters, training input, and labels. A grey-box attack indicates that only limited information about the victim model, like training data labels, is available. As the strictest setting, the black-box attack only allows attackers to do queries of samples for output or labels [16].

In graph classification evasion attack, GradArgmax is a white-box attack method, RL-S2V and ReWatt are typical black-box attack methods based on reinforcement learning. In the aspect of the attacker’s goal, the attacks are divided into untargeted attacks and targeted attacks. In an untargeted attack, the attackers expect the victim model to classify the adversarial samples into labels different from their original labels. As a more strict attack goal, targeted attacks attempt to fool the victim model by specifying the output categories of the adversarial samples.

2.3 Poisoning and Backdoor Attacks on GNNs

Besides evasion attacks, another two typical adversarial attacks on GNN models are poisoning attacks and backdoor attacks. The poisoning attacks [32] suppose the attackers own the ability to poison training data and label. In this way, the performance of GNN models trained on poisoned data will degrade dramatically on clean samples. In backdoor attacks [33], [34], attackers inject a fixed or adaptive trigger into clean training data and change their labels to the desired categories. As a result, the models trained on these data performs well on the clean samples but predicts the desired labels once the well-designed trigger is injected into the clean samples.

3 PRELIMINARY

3.1 Graph Classification

Assuming \( G = \{ V, E \} \) is a graph, where \( V = \{ v_1, \ldots, v_N \} \) is the set of nodes, \( E = \{ e_1, \ldots, e_E \} \) is the set of edges of graph \( G \). The edges set \( E \) describes the structural information of \( G \). It can also be expressed as the adjacency matrix \( A \in \{0, 1\}^{V \times V} \), where \( A_{ij} = 1 \) means the existence of the edge from \( v_i \) to \( v_j \), otherwise \( A_{ij} = 0 \). The features associated with nodes are expressed as matrix \( X \in \mathbb{R}^{V \times d} \), where the \( i \)th row of \( X \) is the features of node \( v_i \) and \( d \) is the dimension of features. So a graph can also be expressed as \( G = \{ A, X \} \).

In the graph classification, a set of graphs is denoted by \( \mathcal{G} = \{ G_i \}_{i=1}^N \) and a label \( y_i \in \mathcal{Y} = \{ 1, 2, \ldots, Y \} \) is associated with each graph \( G_i \), where \( N \) is the number of graphs and \( Y \) is number of categories. The dataset \( \mathcal{D} = \{ (G_i, y_i) \}_{i=1}^N \) is composed of pairs of graph and its label. In graph learning, the classifier \( f \in \mathcal{F} : \mathcal{G} \to \mathcal{Y} \) is trained and expected to learn the mapping from graph \( G \) to its label \( y \) with optimal parameters \( \theta \) that minimize the below loss function

\[
\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} L(f_\theta(G_i), y_i),
\]
where \( L(\cdot, \cdot) \) is used to measure the distance between the predicted and ground-truth labels. A general instance of \( L(\cdot, \cdot) \) is the cross-entropy function.

### 3.2 Graph Neural Networks Model

The graph neural networks (GNNs) is a family of architectures of neural networks that are designed to process graph data \( G = \{V, E\} \). These models iteratively update the expression of nodes by message passing and aggregation as below [35]

\[
m^{t+1}_v = \sum_{w \in N(v)} M_t(h^t_v, h^t_w, e_{vw}), \quad (2)
\]

\[
h^t_v = U_t(h^t_v, m^{t+1}_v), \quad (3)
\]

where \( N(v) \) is the set of neighbours of node \( v \) in graph \( G \), \( h^t_v \) means the hidden expression of node \( v \) at time \( t \in \{1, 2, \ldots, T\} \) and \( e_{vw} \) is the features of the edge from node \( v \) to node \( w \). \( M_t(\cdot, \cdot, \cdot) \) is the message function and \( U_t(\cdot, \cdot) \) is the vertex expression updating function. After the message passing phase, the expression of whole graph \( G \) is obtained by the readout operation

\[
h_G = R(\{h^T_v | v \in G\}), \quad (4)
\]

where \( R(\cdot) \) is the readout operation, and it is invariant to permutations of the nodes. A general instance of \( R(\cdot) \) is the max-pooling or sum-pooling.

In graph classification, the model \( f_\theta \) based on the above architecture is generally trained under an inductive learning setting, in which the classifier learns on training data \( D_{\text{train}} \) and then make graph label prediction on test data \( D_{\text{test}} \).

### 3.3 Problem Formulation

Assuming a GNN model \( f_\theta \) is trained on clean graph samples and then employed to predict the category of graph data \( \mathcal{D} = \{G_j\}_j^M \). In evasion attacks, an attacker \( T \) attempts to make unnoticeable perturbations on the original \( G \) to degrade the performance of this victim model \( f_\theta \). We use \( T_f(G) \) to indicate the attacker’s perturbation on \( G \) which is specifically designed for the classifier \( f_\theta \). In untargeted evasion attacks, attackers expect that the predicted category of an adversarial sample is different from its true category. More precisely, the objective of untargeted evasion attacks on the victim model \( f_\theta \) is

\[
\text{max}_{G} \sum_{j=1}^{M} \mathbb{1}(f_\theta(T_f(G_j)) \neq y_j)
\]

\[
s.t. \quad T_f(G) \quad T_f(G, G; k) = 1.
\]

Here \( \hat{G} \) is the adversarial sample generated from the clean sample \( G \), and \( \mathbb{1}(\cdot) \) is the binary indicator function. \( k \) is the perturbation budget of \( T_f \). Given a specific \( k \), \( T(\cdot; k) \) is a similarity measure function whose output is \( 1 \) when two input graph samples are semantically the same and \( 0 \) otherwise. Particularly, in targeted evasion attacks, attackers expect the predicted category to be a specified category \( y_j \), which is different from its true category \( y_j \). The objective of targeted evasion attacks is

\[
\text{max}_{G} \sum_{j=1}^{M} \mathbb{1}(f_\theta(T_f(G_j)) \neq y_j)
\]

\[
s.t. \quad T_f(G) \quad T_f(G, G; k) = 1.
\]

Note. The probability function \( p(\cdot) \) can be employed as a substitute of \( \mathbb{1}(\cdot) \) to describe objective functions [16]. Given \( G_j \) and budget \( k \), we use \( O(G_j; k) \) to denote the value of objective function in Equations (5) and (6) when using up all budget \( k \) (e.g., in Equation (5), \( O(G_j; k) = \mathbb{1}(f_\theta(G_j) \neq y_j) \) or \( p(f_\theta(G_j) \neq y_j)) \).

### 4 PROPOSED APPROACH

Our projective ranking method mainly includes the ranking module and the projection module. In this section, we first formulate the perturbation space and propose the principles of our evaluation framework. Then we present the detail of the ranking module and the perturbation projection operation. Finally, we give an algorithm to show the generation of adversarial samples.

#### 4.1 Attack Setting

In our method, given a victim model \( f_\theta \), we assume that the attackers can have access to the node embedding and predictive probability distribution of \( f_\theta \) to learn the attack strategy. Then, only accessing the embedding of nodes, the attackers may utilize the learned strategy to attack any samples of the victim model under any budget setting.

To further explore the transferability of the learned strategy (in Section 6.5), we assume an intelligent attacker attempts to fool the victim models by accessing the nodes embedding of the target models based on the attack strategy learned from the other source models engaged in the similar data domain. We note that the attacker does not know any other knowledge (like the architecture or parameters of the target models) except for the embedding of nodes. This scenario reflects the vulnerability of models that use the pre-training models [36].

#### 4.2 Perturbation Space

In an evasion attack, an attacker \( T \) can perturb a clean sample \( G \) to obtain an adversarial sample \( \hat{G} \). We formulate the process of this perturbation on graph \( G \) as

\[
\hat{G} = T_f(G) = G + \Delta G,
\]

(5)

where \( \Delta G = \{\Delta A, \Delta X\} \) is the perturbation graph. To be more precise, the graph structure and node features of \( \hat{G} \) are expressed as

\[
\{\hat{A}, \hat{X}\} = \{A + \Delta A, X + \Delta X\},
\]

\[
\Delta A = m_A \odot ([\text{add} \cdot (I_A - A) + \text{del} \cdot (-A)]),
\]

\[
\Delta X = m_X \odot (I_X - 2X),
\]

(8)

where \( A \in \{0, 1\}^{|V| \times |V|} \) is the adjacency matrix, \( X \in \{0, 1\}^{V \times d} \) is a binary node features matrix, \( I_A = J_\{V\} - I_{|V|} \), \( I_X = J_\{V\} \odot d \) (\( J \) is the all-one matrix, \( I \) is the identity matrix). \(|V| \) is the number of nodes in \( G \), and \( d \) is the size of node features. \( m_A \in \{0, 1\}^{|V| \times |V|} \) and \( m_X \in \{0, 1\}^{|V| \times d} \) represent the masks of
graph structure and node features, respectively. \( \text{I}(\text{add})/\text{I}(\text{del}) \) is the indicator function to show if adding/deleting edge operation is allowed in generating adversarial samples. \( \odot \) is the Hadamard product and \( \cdot \) is the scalar multiplication. Therefore, the perturbation graph \( \Delta G = \{\Delta A, \Delta X \} \) is actually defined by the combination of specified structure mask \( m_A \) and feature mask \( m_X \). The similarity measure function in (5) and (6) is refined as

\[
I(G, \tilde{G}) = I(a||m_A||_1 + ||m_X||_1 \leq k),
\]

where \( || \cdot ||_1 \) is the L1 Norm, \( k \) is the budget of perturbations. \( a \) is a scalar coefficient, and it is 1 if \( G \) is a directed graph and \( \frac{1}{2} \) otherwise, since the adjacency matrix of the undirected graph is symmetric.

Given the clean sample \( G \), the perturbation space \( T(G) \) of \( G \) is defined as the set of all possible perturbation graph \( \Delta G = \{\Delta A, \Delta X \} \). The size of a perturbation graph \( \Delta G \) is defined as

\[
|| \Delta G || = a||m_A||_1 + ||m_X||_1 \in \mathbb{Z}^+_0,
\]

where \( \mathbb{Z}^+_0 \) is the set of non-negative integer. In the perturbation space \( T(G) \), the operators \( \odot \) and \( + \) on perturbation graphs are defined as

\[
\Delta G_{\alpha} \odot \Delta G_{\beta} = \{\Delta A_{\alpha} \odot \Delta A_{\beta}, \Delta X_{\alpha} \odot \Delta X_{\beta}\},
\]

\[
\Delta G_{\alpha} + \Delta G_{\beta} = \{\Delta A_{\alpha} + \Delta A_{\beta}, \Delta X_{\alpha} + \Delta X_{\beta}\},
\]

where \( [\cdot] \) means the elements of the input matrix is limited in \([-1, 1]\). The perturbation space \( T(G) \) with budget \( k \) is defined as

\[
T_k(G) = \{\Delta G \mid || \Delta G || = k\}.
\]

Specially, we call \( \Delta G \in T_k(G) \) an operation element of perturbation space \( T(G) \).

Given the above definitions and operators, the perturbation space \( T(G) \) owns the following properties:

**Property 1.** If \( \Delta G_0 \in T_0(G) \), then \( \forall \Delta G_{\alpha} \in T_m(G) \), we have

\[
\Delta G_0 \odot \Delta G_{\alpha} = \Delta G_0,
\]

\[
\Delta G_0 + \Delta G_{\alpha} = \Delta G_{\alpha}.
\]

**Property 2.** Given \( \Delta G_{\alpha}, \Delta G_{\beta} \in T_1(G) \), \( \Delta G_{\alpha} \odot \Delta G_{\beta} \in T_1(G) \) if \( \alpha = \beta \), otherwise \( \Delta G_{\alpha} \odot \Delta G_{\beta} \in T_0(G) \).

**Property 3.** Given \( \Delta G_{\alpha} \in T_1(G) \), \( \Delta G_{\beta} \in T_{k-1}(G) \) \((k \in \mathbb{Z}^+)\), \( \Delta G_{\alpha} + \Delta G_{\beta} \in T_k(G) \) if \( \Delta G_{\alpha} + \Delta G_{\beta} \in T_0(G) \).

**Property 4.** \( \forall \Delta G_{\alpha} \in T_m(G) \), \( \Delta G_{\beta} \in T_n(G) \), we have \( \Delta G_{\alpha} + \Delta G_{\beta} \in T_{m+n-||\Delta G_{\alpha} \odot \Delta G_{\beta}||}(G) \).

**Proof.** Given the definition of \( T_k(G) \), the result is a direct consequence of the following calculations. Based on the definitions of \( + \) and \( \odot \) operations, we have

\[
|| \Delta G_{\alpha} + \Delta G_{\beta} || = || \Delta G_{\alpha} || + || \Delta G_{\beta} || - || \Delta G_{\alpha} \odot \Delta G_{\beta} ||
\]

\[
= m + n - || \Delta G_{\alpha} \odot \Delta G_{\beta} ||.
\]

Based on Property 3, it is easy to obtain that the perturbation space \( T_k(G) \) is actually composed of \( k \) different operation elements of \( T(G) \), which means that

\[
T_k(G) = \{\Delta G \mid \Delta G = \sum_{k=1}^{k} \Delta G_{\alpha} \odot \Delta G_{\beta} \in T_1(G),
\]

\[
\Delta G_{\alpha} \odot \Delta G_{\beta} \in T_0(G), \ \forall k \neq k_j\},
\]

and the size of \( T_k(G) \) is

\[
|T_k(G)| = C_k^E, \ E = |T_1(G)|,
\]

where \( | \cdot | \) is the size of a set, and \( C \) is the combination operation of two integers.

### 4.3 Projective Ranking

#### 4.3.1 Design Goals and Principles in Evaluation Framework

Incorporating with the objective in (5) and (6), we also consider a fundamental expectation of the adversary as a design goal of evasion attack methods: An intelligent attacker always expects to choose the perturbation operation which will bring the most significant attack benefit at each perturbation step for achieving stealthy and effective attacks.

Assuming that \( O(G; k) \) (see Section 3.3) results from the collaboration of \( k \) different \( \Delta G_i \), where \( \Delta G_i \in T_1(G) \) and \( \sum_{i=1}^{k} \Delta G_i \in T_k(G) \). \( B(\cdot) \) is the attack benefit function defined on \( \Delta G_i \), \( B(\Delta G_i) \) presents the contribution of \( \Delta G_i \) to \( O(G; k) \). The intelligent attackers expect to ranking these \( k \) different \( \Delta G \), w.r.t. \( B(\Delta G_i) \), and at each perturbation step they choose \( \arg\max_{\Delta G} B(\Delta G_i) \). Moreover, the objective in (5) and (6) can also be regarded as ranking all operations in the perturbation space and selecting \( k \) operations. Therefore, the core question in this paper is how to rank \( \Delta G \in T_1(G) \) concerning their relative importance to the attack goal.

**A. From Design Goals to Evaluation Framework**

To achieve the above expectation, we propose an evaluation framework to reveal its core requirements. Our framework is composed of three principles, they are:

1. **Benefit Decomposition.** Explaining \( B(\cdot) \) is necessary for understanding the three principles. When budget \( k > 1 \), perturbations at each step are combined together to function as a whole. Although final attack results come from collaboration of \( k \) different \( \Delta G \) \( \in T_1(G) \), only one \( \Delta G \) \( \in T_1(G) \) can be chosen at each perturbation step. This fact requires that the attack benefit \( B(\cdot) \) should be felicitously defined on \( \Delta G \in T_1(G) \) from a global view, i.e., distributing \( O(G; k) \) to each \( \Delta G \) to support the intelligent attacker making perturbation decision. For example, when using Shapley value \( \phi \) [37] as the contribution distribution function, \( B(\Delta G) = \phi(\Delta G, O(G; k)) \). In this paper, an attack method satisfies the benefit decomposition principle if it considers all perturbations as a whole to evaluate their attack results, and uses some mechanisms to distribute the overall benefit to the perturbations at each step.

2. **Baseline Graph.** When defining \( B(\cdot) \), an implicit ground object is the baseline graph of attack benefit should be fixed on the clean graph \( G \), i.e., \( B(\cdot) = B(\cdot, G) \), and \( B(\Delta G) = 0 \) where \( || \Delta G || = 0 \). If the baseline graph changes to other graphs, the measure of attack benefit will deviate from the original meaning (see the example of \( \text{GradArgmax} \) in Appendix B), which can be found on the Computer Society
Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2022.3219209.

(3) Operation Ranking. To achieve stealthy and effective evasion attacks, given specific values of $B(\cdot)$ on $\Delta G \in T_1(G)$, an intelligent attacker should choose the perturbation with maximal attack benefit in each step. In the above process, the intelligent attacker ranks $\Delta G \in T_1(G)$ according to their attack benefits, and then choose top-$k$ different perturbation operations one-by-one. In this paper, an attack method satisfies the operation ranking principle if its perturbation at each step consumes the attack budget by considering the $B(\cdot)$-based ranking of $\Delta G \in T_1(G)$.

B. From Evaluation Framework to Our Method

In this section, we utilise principles in the evaluation framework and additional requirements to guide the design of our method. Generally, an intelligent attacker always attempts to maximise the profits in attacks (see Appendix A), available in the online supplemental material. In this paper, we extend the definition of transferability of attack strategies. The transferability means that once attackers learn the attack strategies, they can generate adversarial samples for unseen clean samples under different perturbation budgets. However, if attack strategies are learned from fixed budget $k$ (e.g., RL-S2V), these strategy cannot easily satisfy transferability since they are designed for specific $k$. Therefore, the evaluation framework and desired transferability practically require an attack method accomplishes $B(\cdot)$-based ranking of $\Delta G \in T_1(G)$ without considering specific $k$ in $B(\cdot)$.

Inspired by the recent research on GNN explainability [38], [39], we employ all perturbation operation $\Delta G \in T_1(G)$ of the clean graph $G$ (i.e., baseline graph) as a whole to attack target GNNs for getting rid of the limitation of specific $k$. Based on mutual information, the $B(\cdot)$ of each $\Delta G$ is then calculated as their importance (see more in Appendix B) for final attack results, available in the online supplemental material. In this way, our method satisfies the benefit decomposition principle. To satisfy the operation ranking principle, we first rank $\Delta G \in T_1(G)$ according to their importance values (i.e., attack benefit), and then project this ranking into practical attacks with specific $k$.

4.3.2 Ranking Method

Now, given a clean sample $G$ as the baseline graph, attackers attempt to rank all operation elements in perturbation space $T(G)$ with respect to their importance. We measure the importance of elements in $T_1(G)$ by mutual information (MI) between the modified graph $\tilde{G}$ and the attacker’s goal. It can be expressed as

$$\max MI(\tilde{Y}, \tilde{G}) = H(\tilde{Y}) - H(\tilde{Y}|\tilde{G}).$$

(15)

where $H(\cdot)$ is the entropy function. In untargeted evasion attacks, $\tilde{Y} = (\ldots, p_{y_G} = 0, \ldots)$ is the expected prediction distribution of $\tilde{G}$, $y_G$ is the original category of $G$. For the clean sample $G$, the first item in (15) is a constant since the victim model $f_0$ is fixed and $\tilde{Y}$ is also invariant. So the objective in (15) is equal to

$$\min H(\tilde{Y} | \tilde{G}).$$

(16)

To reduce the computational difficulties caused by the discrete graph structure, we apply continuous relaxation on $\Delta G$ and assume it is a graph variable $\Delta G \sim T_1(G)$. Based on Jensen’s inequality and $H(\cdot)$ is a concave function, we obtain that

$$H(\tilde{Y} | \tilde{G}) = \sum_{\Delta G \sim T_1(G)} p(\Delta G) H(\tilde{Y} | G + \Delta G)$$

$$= E_{\Delta G} \left[H(\tilde{Y} | G + \Delta G)\right]$$

$$\leq H(\tilde{Y} | G + E(\Delta G)).$$

(17)

where $E[\cdot]$ indicates expectation function. So the objective (16) is equal to

$$\min_{T_1(G)} H(\tilde{Y} | G + E(T_1(G)) \Delta G).$$

(18)

To be more specific, given the victim model $f_0$, the objective of untargeted evasion attacks is

$$\min_{T_1(G)} \sum_y \mathbb{I}(y \neq y_G) \log p(y | \tilde{G}).$$

(19)

where $p(\cdot | \tilde{G})$ is the predictive probability distribution of $f_0$ on $\tilde{G}$. The objective of targeted evasion attacks is

$$\min_{T_1(G)} \sum_y \mathbb{I}(y = y_t) \log p(y | \tilde{G}).$$

(20)

where $y_t$ is the specified target category.

Following the attack setting in previous methods [22], we assume that attackers are only able to add edges when generating $\tilde{G}$. So we have

$$\tilde{G} = T_1(G) = \{A + \Delta A, X\},$$

(21)

where $\Delta A = m_A \odot (I_A - A)$, the mask $m_A$ is obtained by

$$m_{A_{i,j}} = s(h_i, h_j) = \text{softmax}(\text{MLP}(\text{concat}(h_i, h_j)))$$

(22)

where $h_i$ is the embedding of node $i$, $s(\cdot, \cdot)$ is the scoring function which measures the importance of elements in $T_1(G)$ with respect to the attacker’s expectation. In our method, after concatenating the embedding of nodes $v_i$ and $v_j$, a Multi Layer Perceptron (MLP) model and the softmax function are employed to obtain the final score. Since the value of $\Delta A$ is a real number rather than a discrete value, we call that $\tilde{G}$ is generated by adding continuous structure perturbations $\Delta G = \{\Delta A, 0\}$.

4.3.3 Perturbation Projection

In our method, the learned attack strategy is embedded in the ranking of $\Delta G = \{\Delta A, \Delta X\} \in T_1(G)$. Due to the discrete structure limitation of graph data, although attackers could obtain the adversarial sample $\tilde{G}$ in (21) by adding the continuous perturbation $\Delta G$, it is necessary to bridge the gap between continuous perturbation $\Delta G$ and discrete modification $\Delta G_{\text{dis}}$ under limited budgets. Inspired by Projected Gradient Descent (PGD) algorithm, we map $\Delta G$ to $\Delta G_{\text{dis}}$ by below projection function
Algorithm 1. Generation of Adversarial Samples

Input: $D = \{G^m = (A^m, X^m)\}_{m=1}^{M}$, classifier $f_0$

Output: $\hat{D} = \{G^m = (A^m, X^m)\}_{m=1}^{M}$

Parameters: $\varphi$ // The trainable parameters of $s$

Initialization: $h = \text{NodeEmbed}(D; f_0)$, $\text{ASR}_{\text{best}} = 0, k = 1$

1 $\textbf{Def}$ Ranking $D$:
2 \hspace{1em} for ($m = 1; m <= M; m = m + 1$)
3 \hspace{1em} \hspace{1em} $S^m_{ij} = s_{\varphi}(h^m_{ij}, h^m_{ji})$
4 \hspace{1em} \hspace{1em} $\Delta A^m_{ij} = (I_A^m - A^m) \odot S^m$
5 \hspace{1em} \hspace{1em} return $\{(\Delta A^m_{ij})_{m=1}^{M}\}$
6 $h = \text{NodeEmbed}(D; f_0)$, $\text{ASR}_{\text{best}} = 0, k = 1$

7 $\textbf{Function}$ Main:
8 \hspace{1em} while not EarlyStop
9 \hspace{1em} \hspace{1em} // Attacker Training;
10 \hspace{1em} \hspace{1em} $\{(\Delta A^m_{ij}) = \text{Ranking}(D)\}$
11 \hspace{1em} \hspace{1em} $\hat{G}^m = \{(A^m + \Delta A^m_{ij}, X^m)\}$
12 \hspace{1em} \hspace{1em} $\text{min}_p - \sum_{m=1}^{M} \sum_y 1(y \neq y_C) \log p(y | \hat{G}^m)$
13 \hspace{1em} \hspace{1em} // Attacker Evaluation
14 \hspace{1em} \hspace{1em} $\{(\Delta A^m_{ij}) = \text{topk}(\text{Ranking}(D))\}$
15 \hspace{1em} \hspace{1em} $\hat{D} = \{(A^m + \Delta A^m_{ij}, X^m)\}$
16 \hspace{1em} \hspace{1em} // Update the Attack Success Rate
17 \hspace{1em} \hspace{1em} $\text{if } \text{ASR}($\hat{D}; f_0$) > $\text{ASR}_{\text{best}}$ then
18 \hspace{1em} \hspace{1em} $\text{ASR}_{\text{best}} = \text{ASR}_{\text{current}}$

After obtaining the adversarial samples $\hat{D} = \{\hat{G}^m\}_{m=1}^{M}$, the attacker use them to attack the victim model $f_0$. Algorithm 1 shows how attackers generate adversarial samples from clean samples $D$ by our projective ranking method. To learn an attack strategy, in line 6, attackers first need to obtain the embedding representation of all nodes $h$ of clean samples in $D$ from the victim model $f_0$. In the training phase (lines 9-12), attackers use the scoring function in Equation (22) to obtain structure mask $m_A$, then obtain the perturbations on graph structure $\Delta A$, by limiting $m_A$ with the allowed perturbation operation type. Then, in line 11, attackers add perturbations $\Delta A$ to clean samples to generate a relaxed adversarial sample $\hat{G}_d$. In line 12, attackers use the objective (19) to learn an attack strategy embedded in the scoring function. In the evaluation phase (lines 13-18), attackers will obtain the practical adversarial sample $\hat{G}_d$ to evaluate the attack performance of the learned attack strategy. In line 14, attackers project the learned ranking into the practical attack space, in which attackers are only allowed to add edges with the specified budget ($k = 1$). Then attackers obtain the final adversarial samples in line 15 and attack the victim model in line 17. Attackers will repeat the process (lines 9-18) until the projective ranking method learned a high attack strategy. By replacing the objective in line 12 with objective (20), attackers could obtain adversarial samples in the context of targeted attacks.

5 Method Analysis Under Our Evaluation Framework

In this section, we use the three principles in Section 4.3.1 to make an evaluation on our method and three typical attack methods: RandomSampling, GradArgmax and RL-S2V. The results are summarized in Table 2. In these methods, RandomSampling only satisfies the baseline graph principle, our method satisfies the attack benefit, operation ranking and baseline graph simultaneously. The other methods are partially consistent with the evaluation framework. The reasons are as follows.
TABLE 2
Consistency With the Principles of the Evaluation Framework

| method       | benefit decomposition | baseline graph | operation ranking |
|--------------|-----------------------|----------------|------------------|
| Random       | ●                     | ●              | ●                |
| GradArgmax   | ●                     | ●              | ●                |
| RL-S2V       | ●                     | ●              | ●                |
| Our          | ●                     | ●              | ●                |

○ means neglect, ● means consideration.

(1) GradArgmax. Similar to the use of gradient descent for training in neural networks, GradArgmax employs the gradient of \( f_\theta \) on all operation elements \( \Delta G_{cur} \in T_k(G_{cur}) \) to choose perturbation operation, where \( G_{cur} \) is the graph waiting to be modified at the current perturbation step. Hence, GradArgmax neglects the baseline graph principle. To use up all perturbation budget \( k \), attackers need to calculate the gradient information at each perturbation step. Then it will use the real-time local gradient as the measure of attack benefits. Actually, the gradient information is a measure of modification sensitivity (see Appendix B), available in the online supplemental material, so GradArgmax does not satisfy the attack benefit principle. To generate adversarial samples, GradArgmax focuses on integrating all perturbation graph \( \Delta G_{cur} \in T_k(G_{cur}) \), which owns the biggest local benefit at each perturbation step. Although the operations in GradArgmax have an order, it does not satisfy the operation ranking principle since the local gradient information is not a suitable measure for attack benefits.

(2) RL-S2V. In the generation of adversarial samples, RL-S2V uses the attack benefit of \( \Delta G \in T_k(G) \) to decide the perturbation operation \( \Delta G \in T_k(G) \) of each step in consuming perturbation budget \( k \). The reward function in RL-S2V is designed by considering the outputs of the victim model \( f_\theta \) on \( \tilde{G} \) and \( G \), so it satisfies the baseline graph and attack benefit requirements. However, it emphasizes using \( \Delta G \in T_k(G) \) as a whole perturbation to attack the model \( f_\theta \) with a specified budget \( k \), while cares less about the order of the element \( \Delta G \in T_k(G) \) that composes the perturbation space \( T_k(G) \). Hence, the RL-S2V attack method neglects the attack ranking.

Furthermore, we analyze the difference between the measures of attack benefits (sensitivity, long-term benefit, importance) in the Appendix B, available in the online supplemental material. We also show the weakness of GradArgmax concerning the attack benefits from three different aspects and empirically demonstrate them in our experimental results.

6 EXPERIMENTS

6.1 Datasets and Baselines
We employ several real-world datasets ENZYMES, Mutagenicity, PC-3, NCI109, NCI-H23H [40] and one synthesized dataset BA-2Motifs [39] for graph classification. For the NCI-H23H dataset, we only choose the graphs whose node size is less than 50. Table 3 shows some basic statistics of these datasets.

To evaluate the performance of our method, we select RandomSampling, GradArgmax, and RL-S2V [22] as baselines. In the RandomSampling method, we performed ten times with different seeds and then averaged these accuracy results. In our experiments, the attack performance is measured by the accuracy of the victim model, in which a low accuracy number indicates a high attack performance.

6.2 Experimental Setup
In the victim models, we randomly split each dataset into training data (80%), validation data (10%), and test data (10%) to train the victim GAT [41] or GCN [42] models. The victim models have three hidden layers, with 20 as the output feature dimension. We concatenate the maxpooling and sumpooling results of the final embedding of nodes to obtain the expression of the whole graph, then use a fully connected layer to predict the category of the graph. In our projective ranking method, we use a 2-layer MLP to serve as the scoring module.

To evaluate the transferability of our method, we directly project the learned strategy under \( k = 1 \) to the generation of adversarial samples under \( k = 2 \), \( 3 \). The other results are obtained by running corresponding attack methods under a specified budget, respectively.

6.3 Attack Performance Comparison

6.3.1 Untargeted Attacks
The upper half of Tables 4 and 5 show the accuracy of the victim model \( f_\theta \) on both clean and adversarial samples generated by different attack methods. The bold numbers indicate the best attack results under the same setting.

In Tables 4 and 5, our method achieves the best or competitive attack performance on all datasets with \( k = 1 \), indicating the projective ranking method owns powerful attack performance. When \( k = 2 \) or 3, we use the learned attack strategy under \( k = 1 \) to generate adversarial samples for the changed perturbation budget. The results show that our method also obtains the best or competitive attack effect, indicating the projective ranking method has learned an effective transferable attack strategy under budget \( k = 1 \).

GradArgmax generates the adversarial examples based on the gradient of the victim model and makes a greedy choice at each step. The attack results of GradArgmax in the Tables 4 and 5 empirically prove our discussion about sensitivity in Appendix B, available in the online supplemental material. First, the perturbation generated by GradArgmax is sub-optimal since the gradient is the sensitivity measure of the victim model. Although the GradArgmax achieves effective attack performance on the victim GCN model, its ability on the victim GAT model is limited (see Fig. 2). Second, the result on the NCI109 dataset with \( k = 1 \) in Table 5 shows that GradArgmax generates samples with negative attack
performance. This indicates that its attack performance is limited by the non-linear nature of the GNN model in some situations. Finally, the attack performance of GradArgmax is worse than the RandomSampling on ENZYMES, PC-3 and NCI109 datasets when $k = 2$ and 3, on which GradArgmax shows weak marginal attack performance concerning the budget. Moreover, GradArgmax does not learn any attack strategy in the attack process. Above results indicate that GradArgmax is not desirable for attackers in some situations since it misses the baseline graph and attack benefit principles in evasion attacks.

Compare with using the “local” gradient in GradArgmax, RL-S2V and our method achieve stable attack performance on both GAT and GCN models since both of them consider the importance of perturbation operation with “global” criteria. We also use RandomSampling as a baseline to evaluate other attack methods. Tables 4 and 5 show our method always achieve better attack performance than RandomSampling. The italic shows unexpected results in the aspect of attack ability, where the performance of RL-S2V is worse than RandomSampling on ENZYMES ($k = 2,3$) in Table 4 and ENZYMES ($k = 3$), NCI109 ($k = 3$) in Table 5. Moreover, attackers always expect to obtain better attack performance when giving more budgets. If the budget increases, a well-designed attack method should not decrease its performance. However, like the result in [24], we also observed that the attack performance of RL-S2V degrades when increasing the perturbation budget on Mutagenicity ($k = 3$) in Table 4. This may be due to the less effectiveness of the Q-learning method in RL-S2V, especially for the Markov decision process with a long horizon.

### 6.3.2 Targeted Attacks

We also conduct targeted attacks on the ENZYME dataset. Without loss of generality, these methods attempt to attack GNN models so that they classify graphs from other classes.
The lower half of Tables 4 and 5 show the classification accuracy of $f_{\text{g}}$ on both clean and adversarial samples. We consider the RL-S2V and our projective ranking method since only they can learn transferable attack strategies. The bold results indicate the best method under the same setting, and italic show unexpected results in the aspect of transferable attack performance. First, the results in Tables 4 and 5 show both RL-S2V and our method could use the learned attack strategy to generate effective adversarial

### Table 7

**Accuracy (%) Comparison of the Victim Models without/with Defence**

| Dataset     | ENZYMES | Mutagenicity |
|-------------|---------|--------------|
| $k$         | 1 | 2 | 3 | 1 | 2 | 3 |
| W/0*        | 69.17 | 69.17 | 69.17 | 85.25 | 85.25 | 85.25 |
| Ours        | 52.50 | 44.79 | 42.92 | 67.63 | 59.13 | 53.63 |
| W $* \beta = 0.9$ | 22.29 | 22.29 | 22.29 | 53.00 | 53.00 | 53.00 |
| Ours        | 22.29 | 22.29 | 22.29 | 52.37 | 52.13 | 52.00 |
| W $\beta = 0.7$ | 17.50 | 17.50 | 17.50 | 49.13 | 49.13 | 49.13 |
| Ours        | 17.50 | 17.50 | 17.50 | 49.25 | 49.00 | 49.00 |

*W/o indicates results on vanilla GCN models, and W indicates results on GCN models equipped with randomized smoothing. $\beta$ presents the noise parameter in the randomized smoothing.*

GAT model on the NCI-H23H dataset, we employ undersampling on the category with most samples to alleviate the unfavourable effect caused by unbalanced distribution. In the training data, the # of positive samples: # of negative samples = 1:2.

All results in this section are achieved by directly applying attack methods on the vanilla GNN models. Recently, the randomized smoothing method [44], [45] has been widely employed to improve the robustness of AI models. In this paper, we follow the setting (noise parameter $\beta = 0.9/0.7$, sampling number $d = 10000$) of recent research on certified robustness of GNNs [46] to evaluate the attack performance of our method. Experimental results show randomized smoothing [46] is an effective defence method. However, for GNN models in this paper, randomized smoothing is not applicable because it significantly degrades the performance (i.e., accuracy) of GNNs. For example, as shown in Table 7, although our method almost cannot attack the GCN models on ENZYMES/Mutagenicity after introducing randomized smoothing, the accuracy of GCN equipped with the defence method ($\beta = 0.9$) on clean graphs is only $22.29% / 53.00\%$, which is dramatically lower than that of vanilla GCN models suffering attacks with $k = 1$ (i.e., $52.50% / 67.63\%$).

### 6.4 Transferability on Unseen Samples

To reduce attackers’ effort to generate adversarial samples from unseen clean samples, they generally expect a transferable attack strategy. In this section, we use the attack strategies obtained in Section 6.3 to generate adversarial samples for unseen samples and then employ them to attack the victim models.

#### 6.4.1 Untargeted Attacks

The lower half of Tables 4 and 5 show the classification accuracy of $f_{\text{g}}$ on both clean and adversarial samples. We consider the RL-S2V and our projective ranking method since only they can learn transferable attack strategies. The bold results indicate the best method under the same setting, and italic show unexpected results in the aspect of transferable attack performance. First, the results in Tables 4 and 5 show both RL-S2V and our method could use the learned attack strategy to generate effective adversarial
samples under perturbation budget \( k = 1 \) on all datasets. It indicates that the learned strategy can be transferred to attack unseen samples. Second, the results under \( k = 2, 3 \) further demonstrate that the projective ranking method owns transferable attack performance on unseen samples when the budget changes. Especially, our method obtains the best attack performance on some datasets when \( k = 2 \) or 3 even if it does not own the best performance when \( k = 1 \). Finally, compared to RandomSampling, the transferability of the attack strategy of RL-S2V is limited on most datasets when \( k > 1 \). These results reveal that the victim models make consistent mistakes at both the sample and the budget levels. The attackers can utilize this fact to learn the attack strategy only under a specific budget with limited resources and then use the attack strategies to attack the target classifiers when the attack scenario changes.

6.4.2 Targeted Attacks
The lower half of Table 6 shows attack success rate of attack methods on unseen samples. Attack results show our method observably outperforms RandomSampling and RL-S2V. Moreover, in targeted attacks, the transferability of attack strategy from RL-S2V is also limited (see italic results) when comparing with RandomSampling.

6.5 Other Transferability Evaluations
6.5.1 Transferability on Attack Budget
Fig. 3 shows the attack performance of our method on seen samples under different attack budgets. The performance curves illustrate our method learned effective and transferable attack strategies in untargeted and targeted attacks. Moreover, Fig. 3 also empirically demonstrates that our method satisfies the operation ranking principle in the evaluation framework since smaller budgets own larger marginal attack performance.

6.5.2 Transferability on Victim Model
To further explore the transferability, we use the adversarial samples generated from GCN/GAT model to attack the victim GAT/GCN model. The attack results are shown in Table 8, and these adversarial samples achieve better attack performance than RandomSampling. It indicates these victim models used for the same task on the same dataset may make consistent mistakes to some extent.

6.5.3 Transferability on Data Domain
This evaluation explores the transferability of the attack strategy learned on one dataset to attack the victim models built on other datasets. We visualized the attack results on the victim GCN model in Fig. 4. First, the relative attack
performance’s colour indicates that when the source attacker and the target model have the same dataset, it achieves the best attack performance. Second, since the graphs in all datasets are small molecules or come from bioinformatics, the result values in Fig. 4 show the domain similarity of these graphs brings non-negative attack performance.

These results reveal the potential risk of employing a pre-training model with a freezing setting [36] in the downstream tasks. In Fig. 4, we observe that when attacking the NCI-H23H dataset using the attack strategy from the ENZYMES dataset, the accuracy performance drops 8.9%, which is larger than that of RandSampling (4.7%) and GradArgmax (8.6%). Once the adversary knows the victim model \( f(\theta) \) uses the pre-training model \( f_{emb} \), they could train a classifier \( f'_{\theta}(\phi) \) on graphs \( G_f \) that come from the same domain of the data \( G_f \) in \( f(\theta) \), then use the attack strategy learned on \( f'_{\theta}(\phi) \) and the embedding of \( G_f \) from \( f_{emb} \) to attack the victim models.

### 6.6 Visualization of Attack Patterns

To reveal the attack strategies of the projective ranking method, we visualize the perturbations in adversarial samples on the BA-2Motifs dataset based on the GCN model. BA-2Motifs is a synthetic dataset in which two motifs (House and Pentagon) are attached on random base graphs. Table 9 shows the attack results of different methods.

In untargeted evasion attacks, we found two fundamental strategies in the generation of adversarial samples. They are: (1) placing the adversarial samples in the high-confidence region of the other categories in the victim models, and (2) moving the clean samples to the classification boundary or other low-confidence areas of the victim models. We dive into the adversarial samples and observe some enlightening attack patterns in them:

- **Imitation of Other Motifs.** In the first row of the first block in Fig. 5, both RL-S2V and our method add one edge in the Pentagon so that there is a House in the adversarial samples. In this way, the victim GCN model classifies the adversarial samples as House graphs.

- **Collapse of Self-Motif.** In the second row of the first block and the second block in Fig. 5, both RL-S2V and our method attack the victim model by adding one edge in which one node is in the self-motif. In this way, the attackers destroy the self-motif pattern by moving the samples to the low-confidence areas to fool the victim model.

Although the attack results in Table 9 indicate RL-S2V and our method have the same attack performance, we find the attack patterns of RL-S2V are simple and lack variation from the second column in both Figs. 5 and 6. However, another two attack patterns are found in our method:

- **Coexistence of Motifs.** In the first block of Fig. 6, our method shows a new attack pattern. One edge is added

---

**TABLE 8**

Classification Accuracy of the Victim GAT/GCN Model (%)

| Dataset       | ENZYMES | Mutagenicity | PC-3 | NCI109 | NCI-H23H |
|---------------|---------|--------------|------|--------|----------|
| k 1 2 3       | 1 2 3   | 1 2 3        | 1 2 3| 1 2 3  | 1 2 3    |
| Clean         | 65.83   | 65.83        | 65.83| 73.63  | 73.63    |
| Rand          | 60.04   | 51.71        | 45.79| 70.86  | 69.08    |
| GCN-GAT       | 57.50   | 49.38        | 44.17| 71.25  | 65.88    |
| GAT-GCN       | 56.04   | 43.96        | 36.46| 80.63  | 74.13    |

*This means we use the adversarial sample generated from the GCN model to attack the victim GAT model.

---

**TABLE 9**

Classification Accuracy of the Victim Model (%) on BA-2Motifs

| k 1 2 3       | current samples | unseen samples |
|---------------|-----------------|----------------|
| Clean         | 99.63           | 99.63          | 100.00 | 100.00 | 100.00 |
| Rand          | 78.85           | 54.35          | 50.09  | 81.00  | 55.35  |
| Grad          | 60.50           | 51.13          | 50.13  | -      | -      |
| RL-S2V        | 50.00           | 50.00          | 50.00  | 50.00  | 50.00  |
| Ours          | 50.00           | 50.00          | 50.00  | 50.00  | 50.00  |
between two nodes that are outside of the self-motif in the clean samples. The exciting part of this edge is that the distance of the two end-nodes is 5, and after this perturbation operation, there is a motif Pentagon in these samples. The coexistence of motifs is regarded as one behaviour that belongs to the second strategy in generating adversarial samples, in which the clean samples are moved to the decision boundary of the victim models.

**Fake Self-Motif.** In the second block of Fig. 6, our method adds one edge between two nodes outside of the self-motif, resulting in one triangle constructed in the adversarial samples. As a critical part of the House motif, the constructed triangle could confuse the victim models since two “similar” self-motifs exist in current samples. In this way, the samples are moved from the high-confidence region of the victim model to the low-confidence area.

In this section, we visualize the adversarial samples and summarize two basic attack strategies in them. We also found four attack patterns—imitation of other motifs, the collapse of self-motif, coexistence of motifs, and fake self-motif—of the adversarial samples. Compared with the RL-S2V, the adversarial samples generated by the projective ranking method show various attack patterns. These attack patterns are helpful for the adversary to find the graph pattern differences between different data classes. Based on these observations, the attackers can customize the pattern operation to obtain adversarial samples of the specified type of clean graph samples.

### 7 Conclusion

In this article, we present the projective ranking approach to perform an evasion attack for graph classification. We contend that current evasion approaches either do not provide adequate attack performance without considering the perturbations’ long-term benefits or require attackers to readjust the attack strategies when the attack environment changes. To that purpose, we define the perturbation space and propose an evaluation framework on the evasion attacks for graph classification. Then we first relax perturbation space for ranking its elements based on mutual information, and then we project the ranking into generating adversarial samples with a specified budget. The experimental results show that our method performs well in attack performance, and the learned attack strategies can be directly transferred to generate adversarial samples when the budget changes. Furthermore, the visualization of adversarial samples generated by our method shows a variety of attack patterns, which helps identify the vulnerability of the victim models.

Our future work includes simultaneously making perturbations in structure and node features and performing evasion attacks on directed graphs and graphs with edge attributes. In addition, two promising directions are exploring how imbalanced datasets influence the performance/robustness of GNNs and the interaction between robustness and explainability [47] by diving into adversarial samples.

### References

[1] T. Ma, C. Xiao, J. Zhou, and F. Wang, “Drug similarity integration through attentive multi-view graph auto-encoders,” in Proc. 27th Int. Joint Conf. Artif. Intell., 2018, pp. 3477–3483.

[2] S. Fortunato et al., “Science of science,” Science, vol. 359, 2018, Art. no. eaao0185.

[3] C. Zheng, X. Fan, S. Pan, Z. Wu, C. Wang, and P. S. Yu, “Spatiotemporal joint graph convolutional networks for traffic forecasting,” 2021, arXiv:2111.13684.
[4] M. Jin, Y. Zheng, Y.-F. Li, S. Chen, B. Yang, and S. Pan, “Multivariate time series forecasting with dynamic graph neural ODEs,” 2022, arXiv:2202.08408.

[5] X. Zheng, Y. Liu, S. Pan, M. Zhang, D. Jin, and P. S. Yu, “Graph neural networks for graphs with heterophily: A survey,” 2022, arXiv:2202.07002.

[6] M. Jin, Y. Zheng, Y.-F. Li, C. Gong, C. Zhou, and S. Pan, “Multi-scale contrastive siamese networks for self-supervised graph representation learning,” in Proc. 30th Int. Joint Conf. Artif. Intell., 2021, pp. 1477–1483.

[7] M. Wu, S. Pan, and X. Zhu, “OpenWGL: Open-world global learning for unseen class node classification,” Knowl. Inf. Syst., vol. 63, pp. 2405–2430, 2021.

[8] B. Xiong, S. Zhu, N. Potyka, S. Pan, C. Zhou, and S. Staab, “Semi-riemannian graph convolutional networks,” 2021, arXiv:2106.03134.

[9] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, “Anomaly detection on attributed networks via contrastive self-supervised learning,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 6, pp. 2378–2392, Jun. 2022.

[10] Y. Zheng et al., “From unsupervised to few-shot graph anomaly detection: A multi-scale contrastive learning approach,” 2022, arXiv:2202.05525.

[11] J. Gao, J. Gao, X. Ying, M. Lu, and J. Wang, “Higher-order interaction goes neural: A substructure assembling graph attention network for graph classification,” IEEE Trans. Knowl. Data Eng., early access, Aug. 18, 2021, doi: 10.1109/TKDE.2021.3105544.

[12] Z. Liu, Y. Fang, Y. Liu, and V. W. Zheng, “Neighbor-anchoring adversarial graph neural networks,” IEEE Trans. Knowl. Data Eng., early access, Jun. 11, 2021, doi: 10.1109/TKDE.2021.3087970.

[13] P. Li, Y. Wang, H. Wang, and J. Leskovec, “Distance encoding: Design provably more powerful neural networks for graph representation learning,” in Proc. 34th Int. Conf. Neural Inf. Process. Syst., 2020, Art. no. 375.

[14] J. Liu et al., “Shifu2: A network representation learning based model for advising for relationship mining,” IEEE Trans. Knowl. Data Eng., vol. 33, no. 4, pp. 1763–1777, Apr. 2021.

[15] J. Li et al., “Higher-order attribute-enhancing heterogeneous graph neural networks,” IEEE Trans. Knowl. Data Eng., early access, Apr. 23, 2021, doi: 10.1109/TKDE.2021.3074654.

[16] W. Jin et al., “Adversarial attacks and defenses on graphs,” in Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2020, pp. 19–34.

[17] C. Chen et al., “A survey of adversarial learning on graphs,” 2020, arXiv:2003.05730.

[18] B. Wu, X. Yang, S. Pan, and X. Yuan, “Adapting membership inference attacks to GNN for graph classification: Approaches and implications,” in Proc. IEEE Int. Conf. Data Mining, 2021, pp. 1421–1428.

[19] L. Sun et al., “Adversarial attack and defense on graph data: A survey,” 2018, arXiv:1812.10528.

[20] B. Wu, X. Yang, S. Pan, and X. Yuan, “Model extraction attacks on graph neural networks: Taxonomy and realization,” in Proc. ACM Asia Conf. Comput. Commun. Secur., 2022, pp. 337–350.

[21] M. Zhang, X. Wang, M. Zhu, C. Shi, Z. Zhang, and J. Zhou, “Robust heterogeneous graph neural networks against adversarial attacks,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2022, pp. 1399–1407.

[22] H. Dai et al., “Adversarial attack on graph structured data,” in Proc. 35th Int. Conf. Mach. Learn., 2018, pp. 1115–1124.

[23] D. Zügner, A. Akbarnejad, and S. Günnemann, “Adversarial attacks on neural networks for graph data,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2018, pp. 2847–2856.

[24] Y. Ma, S. Wang, T. Derr, L. Wu, and J. Tang, “Graph adversarial attack via rewiring,” in Proc. 27th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2021, pp. 1161–1169.

[25] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, “Hierarchical graph representation learning with differentiable pooling,” in Proc. 32nd Int. Conf. Neural Inf. Process. Syst., 2018, pp. 4805–4815.

[26] B. Xiong, S. Zhu, S. Pan, and J. Tang, “Graph convolutional networks with EigenPooling,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2019, pp. 723–731.
Xingliang Yuan received the PhD degree in computer science from the City University of Hong Kong, in 2016. He is currently a senior lecturer with the Faculty of Information Technology, Monash University, Australia. His research interests include data security and privacy, machine learning security, secure networked systems, and confidential computing. His research has been supported by Australian Research Council, CSIRO Data61, and Oceania Cyber Security Centre. In the past few years, his work has appeared in prestigious venues in information security, computer networks, and distributed systems, such as ACM CCS, NDSS, IEEE INFOCOM, IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Parallel and Distributed Systems, etc.

Chuan Zhou received the PhD degree from the Chinese Academy of Sciences, in 2013. He won an outstanding doctoral dissertation of Chinese Academy of Sciences, in 2014, the Best Paper Award of ICCS-14, and the Best Student Paper Award of IJCNN-17. Currently, he is an associate professor with the Academy of Mathematics and Systems Science, Chinese Academy of Sciences. His research interests include social network analysis and graph mining. To date, he has published more than 90 papers, including IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Pattern Analysis and Machine Intelligence, NeurIPS, AAAI, IJCAI, etc.

Shirui Pan received the PhD degree in computer science from the University of Technology Sydney (UTS), Ultimo, NSW, Australia. He is a professor and an ARC Future fellow (2021-2025) with the School of ICT, Griffith University, Australia. His research interests include data mining and machine learning. To date, he has published extensively in top-tier journals and conferences, including IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Neural Networks and Learning Systems, IEEE Transactions on Knowledge and Data Engineering, ICML, NeurIPS, KDD, AAAI, IJCAI, WWW, CVPR, ACL, and ICDM. He is a recipient of the Best Student Paper Award of IEEE ICDM 2020. He is recognised as one of the AI 2000 AAAI/IJCAI Most Influential Scholars in Australia (2021).

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.