Breaking Free from Label Limitations: A Novel Unsupervised Attack Method for Graph Classification

Yadong Wang[1], Zhiwei Zhang[1]✉, Pengpeng Qiao[2], Ye Yuan[1], Hao Zhang[3], and Guoren wang[1]

[1] Beijing Institute of Technology, Beijing, China
[2] Institute of Science Tokyo, Tokyo, Japan
[3] Huawei Cloud Database Innovation Lab, Beijing, China
{bit.wangyd, peng2qiao}@gmail.com, {zwzhang,yuan-ye}@bit.edu.cn, zhanghao687@huawei.com, wanggrbit@126.com

Abstract. Graph Neural Networks (GNNs) have proven highly effective for graph classification across diverse fields. However, despite their success, GNNs remain vulnerable to adversarial attacks that can significantly degrade their classification accuracy. Existing adversarial attack strategies mainly focus on supervised approaches, limiting their applicability in scenarios where the label information is scarce or unavailable. This paper introduces an innovative unsupervised attack method for graph classification that operates without relying on label information. Specifically, our method first leverages a graph contrastive learning loss to learn robust graph embeddings by comparing different stochastic augmented views of the graphs. To effectively perturb the graphs, we introduce an implicit estimator and flip edges with the top-k highest scores, determined by the estimator, to maximize the degradation of the model's performance. Experiments on multiple datasets show the effectiveness of our proposed unsupervised attack strategy in degrading the performance of various graph classification models.

Keywords: GNNs · Graph Classification · Adversarial Attack

1 Introduction

Graph neural networks have shown outstanding performance across a wide range of real-world applications, including social networks [7], bioinformatics [8], communication networks [12], and chemical structures [2]. The task of graph classification [17,3], aiming to assign entire graphs to predefined categories, has been rapidly developed, providing strong support for various intricate applications.

However, recent research has shown that GNNs are vulnerable to adversarial attacks [16,14], where minor modifications to the graph, such as adding or removing edges or nodes, can cause significant performance degradation. For example, in drug discovery, attackers can alter molecular structures to mislead models into misclassifying toxic molecules as benign, potentially threatening human health [10]. Therefore, graph evasion attacks not only threaten the robustness of the model but could also have severe consequences in real-world applications, highlighting the importance of research in this area.

For adversarial research on GNNs, node-level tasks have received considerable attention, while studies on graph classification tasks are relatively scarce. Current attack methods for graph classification require direct or indirect access to graph label information to guide the adversarial training process. In white-box attacks, the gradient-based Gradargmax [1] method not only relies on label information from the dataset but also needs access to the target GNN's parameters for backpropagation. In black box attacks, it is still necessary to query the model to obtain hard label information (i.e., know the predicted label of the model, but not the prediction probability related to that label). For example, [5] also relies on hard label information to guide its optimization process. RL-S2V [1] and Rewatt [4] based on reinforcement learning require label information to verify the success of the attack and determine the reward value. The dependence of all these methods on label information limits their practical applicability in real-world applications.

Recently, since the labels are hard to acquire, contrastive learning as a novel unsupervised learning framework [6], exhibits performance comparable to supervised learning. However, graph contrastive learning in an unsupervised attack for graph classification is a non-trivial task. Unlike continuous data in images, graphs are discrete, and how to utilize the results of contrastive learning to modify graph data for an effective attack is a challenge. In addition, our goal is to modify the graph so that it can affect the prediction results of downstream tasks. Therefore, an essential challenge is determining how to effectively evaluate the quality of the attacked graph.

To tackle these challenges, we introduce a novel unsupervised graph classification attack strategy based on graph contrastive learning for graph embedding. Specifically, our method first utilizes graph contrastive learning models to extract representations of graphs. We generate different graph views using data augmentation, convert them into embeddings through a graph encoder, and employ a contrastive loss to maximize the similarity between the embeddings of the same graph's views while minimizing the similarity between the embeddings of different graphs' views. Then, we propose an implicit estimator to measure the impact of various possible graph attacks (*insert* or *delete* edges) on graph classification. Finally, we alter the edges with the top-k highest scores from the estimator, effectively attacking the graph and consequently reducing the performance of graph classification.

Our contributions. (1) We introduce an unsupervised method based on graph contrastive learning for conducting adversarial attacks in graph classification tasks. (2) We propose an implicit estimator to measure the effects of graph modifications and poison the graph via the estimator, thus degrading the overall performance of graph classification. (3) Experiments show that our proposed strategy achieves comparable performance with supervised attack methods across four benchmark datasets for graph classification tasks.

2 Related Work

Graph adversarial attacks can be broadly classified into two types: evasion attacks and poisoning attacks [9]. We focus on evasion attacks, as they are more relevant to graph classification tasks. In evasion attacks, the model's training parameters are assumed to remain unchanged. The attacker aims to generate adversarial samples that target the already trained model. Importantly, this type of attack alters only the test data, without necessitating model retraining.

[1] provides a significant starting point for researching evasion attacks on GNNs, introducing the RL-S2V suitable for black-box attacks. However, reinforcement learning-based methods [4] are computationally intensive and often struggle to adapt to varying attack budgets. Diverging from the above two methods, [5] proposes an optimization-based approach in a black-box context, designing a symbolic stochastic gradient descent algorithm with guaranteed convergence. [1] introduces Gradargmax, suitable for white-box attacks, greedily selecting based on gradient information, but its performance in graph evasion attack tasks is average. [15] designs a projection sorting method based on mutual information in the white-box attack scenario.

3 Preliminaries and Problem Formulation

A graph set $\mathcal{G}$ comprises multiple graphs, where each graph G^m is defined as an ordered pair $G^m = (V^m, E^m)$, with V being the nodes set and E being the edges set. For any edge $e_{ij}^m \in E^m$, it connects the node pair (v_i^m, v_j^m), where $v_i^m, v_j^m \in V^m$. For the graph G^m, elements of its adjacency matrix A^m, A_{ij}^m, are 1 if there's an edge between nodes v_i^m and v_j^m, and 0 otherwise. The initial node features are represented by X.

3.1 Preliminaries

Graph Classification. Given a set of graphs $\mathcal{G} = \{G^1, G^2, \ldots, G^M\}$, graph classification aims to assign a class label to each graph by learning a mapping:

$$f : \mathcal{G} \rightarrow \mathcal{C}, \tag{1}$$

where $\mathcal{C}$ is the set of class labels. The optimization objective is to minimize the loss L between predicted and true labels, for each graph G^m with true label y^m:

$$\min_{\theta} \sum_{m=1}^{M} L(f_\theta(G_m), y^m). \tag{2}$$

GNN Models. GNNs update node representations via message passing. A typical GNN layer is:

$$h_v^{(l+1)} = \sigma(W^{(l)} h_v^{(l)} + \sum_{u \in N(v)} W_{\text{rel}}^{(l)} h_u^{(l)}), \tag{3}$$

where $h_v^{(l)}$ represents the embedding of node v at layer l, with the initial node features X serving as $h_v^{(0)}$. $N(v)$ is the set of neighbors of node v, $W^{(l)}$ and $W_{\text{rel}}^{(l)}$ are trainable weight matrices, and σ refers to a non-linear activation function.

After propagating through L layers, a graph-level representation is obtained by aggregating the node-level embeddings through a readout function:

$$H_g = \text{Readout}(\{h_v^{(L)} \mid v \in G\}). \tag{4}$$

Typically, the readout function uses average, sum, or max pooling.

3.2 Problem Formulation

Attacker's Knowledge and Capability. This paper examines evasion attacks on GNNs in a black-box setting, where the attacker accesses only the graph's node features and structure, without knowledge of the target model f or node labels. The attack is confined to modifying the graph's adjacency matrix, enhancing its applicability to graphs lacking node-specific features.
Adversary's Objective. The adversary aims to alter the graph (e.g., *add* or *remove* edges) to degrade classification performance, without access to label information. The objective for a non-targeted evasion attack on model f_θ is:

$$\begin{aligned} \max_{G'} \quad & \sum_{m=1}^{M} \mathbb{I}\left(f_\theta(G'_m) \neq y_m\right) \\ \text{s.t.} \quad & \|G' - G\| \leq \Delta, \end{aligned} \tag{5}$$

where $\mathbb{I}(\cdot)$ is an indicator function, G' is the modified graph, and Δ is the attack budget ensuring modifications remain plausible.

In the absence of labels, we evaluate the similarity of graph embeddings pre- and post-attack using cosine distance, which focuses on directional changes in embeddings rather than magnitude, providing a robust measure of structural alterations. Our attack strategy maximizes this cosine distance between embeddings before and after graph modifications. We train an unsupervised model ϕ using contrastive learning to obtain embeddings, modifying only the graph structure while keeping node features unchanged. The formalized objective is:

$$\begin{aligned} \max_{A'_m} \quad & \sum_{m=1}^{M} dis(f_\phi({A_m}', X_m), f_\phi(A_m, X_m)) \\ \text{s.t.} \quad & \|A' - A\| \leq \Delta, \end{aligned} \tag{6}$$

with cosine distance defined as:

$$dis(f_\phi(A', X), f_\phi(A, X)) = 1 - \frac{f_\phi(A', X) \cdot f_\phi(A, X)}{\|f_\phi(A', X)\| \|f_\phi(A, X)\|}. \tag{7}$$

Values range from 0 (identical) to 2 (maximally different). To achieve this, we train a proxy model via contrastive learning and an implicit estimator to assess the impact of edge modifications. Edges are then modified based on their ranked impact to maximize the cosine distance.

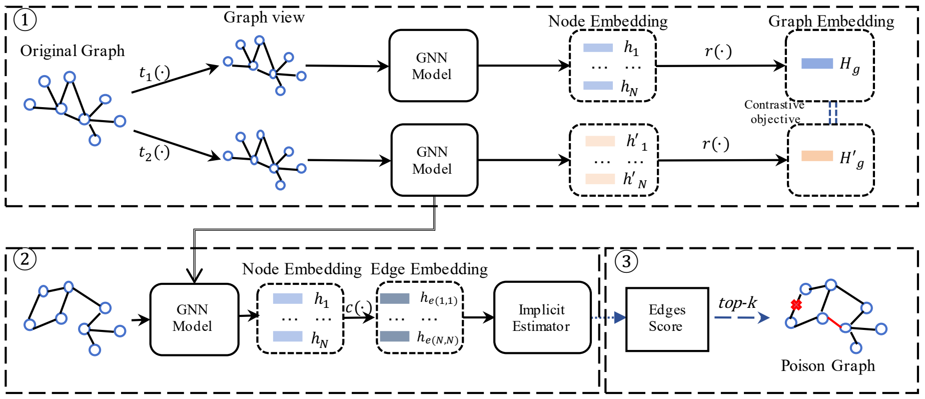


Fig. 1: Attack workflow: Steps ①, ②, ③ correspond to our method's three stages. The function $r(\cdot)$ denotes the Readout operation, $t(\cdot)$ indicates two randomly selected data augmentations, and $c(\cdot)$ signifies the concat operation.

4 Our Design

Our method consists of three stages: proxy model training, attack model training, and attack execution, as shown in Figure 1.

4.1 Proxy Model Training Stage

In this stage, our goal is to train a proxy model capable of obtaining graph representations. We employ a GNN [11] as proxy model. Given that we do not have access to graph label, we employ the graph contrastive learning technique from GraphCL [13] to pre-train GNNs.

During the training phase, for each original graph, we randomly select two out of four possible stochastic augmentation techniques (Node dropping, Edge modification, Subgraph sampling, Feature masking) to obtain two augmented graphs. Then, we feed these augmented graphs into the GNN proxy model with identical parameters to get the graph representations H_a and H_b (defined in Equation 4). Finally, we compare the representations of different augmented graphs and update the GNN parameters according to the contrastive loss, iterating this process for the predetermined number of iterations. The loss function of contrastive learning is formulated as:

$$loss_{pro} = -\log \frac{\exp(\mathrm{sim}(H_{m,a}, H_{m,b})/\tau)}{\sum_{m'=1,m'\neq m}^{M} \exp(\mathrm{sim}(H_{m,a}, H_{m',b})/\tau)}\ , \tag{8}$$

where M represents the total number of graphs; $H_{m,a}$ and $H_{m,b}$ denote the two augmented graph representations of the m-th graph; τ denotes the temperature parameter. The function $\mathrm{sim}(H_{m,a}, H_{m,b})$ quantifies the cosine similarity between the embeddings $H_{m,a}$ and $H_{m,b}$. This cosine similarity function is defined as $\mathrm{sim}(H_{m,a}, H_{m,b}) = \frac{H_{m,a}^{\top} H_{m,b}}{\|H_{m,a}\|\|H_{m,b}\|}$. The notation $\|\cdot\|$ indicates the L_2 norm.

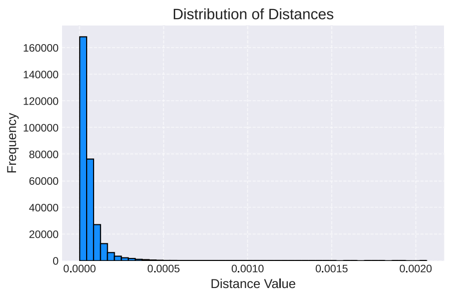


Fig. 2: the distribution of d_{ij} from an experiment on NCI1 dataset.

4.2 Attack Model Training Stage

At this stage, we propose an implicit estimator to evaluate the impact of edge modifications on the graph representations. This impact is quantified by calculating the cosine distance between graph representations before and after the modifications. For each potential edge between node pairs within a graph, we construct each edge embedding by concat h_i and h_j where the embeddings of each node pairs generates from GNN proxy model (defined in Equation 3).

$$h_{e(i,j)} = \text{concat}(h_i, h_j). \tag{9}$$

This edge is then flipped—if it exists, the edge is removed; if not, the edge is added—resulting in an attacked graph (A', X). The attacked graph is fed into the GNN model, as trained in 4.1, to obtain the attacked graph embedding H'_g.

We compute the cosine distance d_{ij} between the original graph embedding H_g and the embedding of the graph post-attack H'_g as the ground truth label for our estimator, as defined in Equation 7. The estimator then employs a MLP model to approximate this ground truth:

$$\hat{d}_{ij} = MLP_{\psi}(h_{e(i,j)}). \tag{10}$$

Here, $\hat{d}_{ij}$ is the estimator's predicted value, which also signifies the estimated importance of the edge $e(i,j)$, and ψ encapsulates the parameters of the MLP model. To this end, we iteratively train the estimator for a specified number of epochs to ensure convergence. Subsequently, the trained estimator is employed to execute the attack operations as detailed in Section 4.3.

In our study, we employ GNN model trained from contrastive learning to generate graph representations.

As shown in Figure 2, we present the distribution of cosine distances d_{ij} between graphs before and after attack. We can observe that although modifications to single edge have a minimal impact on the overall cosine distance, with the maximum value being a mere 0.002 (located at the far right of the axis), there are significant variations in the impact of different edge modifications on the cosine distance.

Due to flipping only one edge at a time, the cosine distance values before and after the graph modification are small. To ensure normal training of the neural network, we scale all cosine distances by a specific coefficient for effective neural network training.

Therefore, the loss function of estimator is computed by:

$$loss_{att} = \sum ||\lambda * d_{ij} - \hat{d}_{i,j}||^2. \tag{11}$$

4.3 Attack Execution Stage

In the third stage, we use the trained estimator to predict the impact of modifying various edges on the overall graph features, aiming to execute an unsupervised attack by flipping the most critical edges. Specifically, for each graph G, we rank all potential edges $e(i,j) \in e$ (whether existent or not) based on the predicted value $\hat{d}_{i,j}$, computed by the estimator trained in Section 4.2. Given an attack budget K (i.e., Δ), we select the top K edges with the highest scores:

$$\begin{aligned} E_{\text{attack}} &= \{e_1, e_2, \ldots, e_K\}, \\ \text{where } e_i &\in \text{sort}(e), \text{ for } i = 1, 2, \ldots, K. \end{aligned} \tag{12}$$

Here, e_1 is the edge with the highest score, e_2 is the edge with the second-highest score, and so on, up to e_K. Based on E_{attack}, we flip these edges and generate attacked graph, termed poisoned graphs.

4.4 Complexity Analysis

Time complexity: Our time complexity mainly depends on the model used for training, and our model uses a 3-layer GCN and a 2-layer MLP, so the time complexity is not high. **Space complexity:** As we calculate the score for each edge in the graph, the memory requirement is $O(N^2)$, where N denotes the number of nodes. In graph classification tasks, the edge count per graph is generally limited (typically in the tens, not exceeding a few hundred), so this requirement is entirely manageable.

5 Experiments

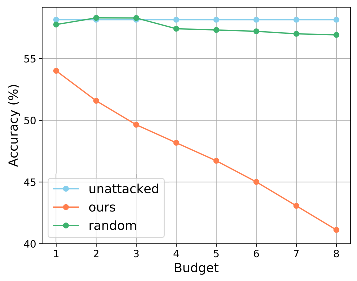
5.1 Experimental Setting

Datasets. We select four public datasets: *NCI1* , *NCI109*, *Mutagenicity* and *ENZYMES*. A summary of these datasets is in Table 1.

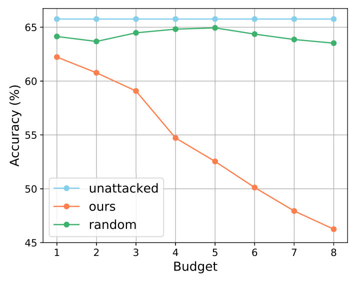
Baseline. (1) **Unattacked**. The GNN model is trained on without attacked graphs. (2) **Random**. The GNN model is trained on poisoned graphs with randomly attacked edges. (3) **GradArgmax [1]**. Select the edge with the largest absolute value of gradient to flip. (4) **Projective Ranking [15]**. Using mutual information, it sorts and evaluates the attack benefits of each disturbance to obtain effective attack strategies.

Table 1: Summary of datasets used.

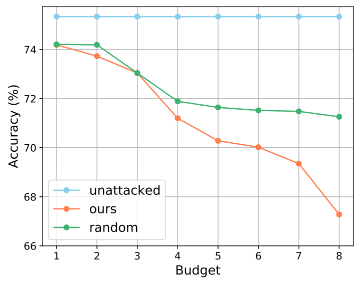
	Mutagenicity	**NCI1**	**NCI109**	**ENZYMES**
Number of Graphs	4337	4110	4127	600
Number of Classes	2	2	2	6
Avg. Number of Nodes	30.32	29.87	29.68	32.63
Avg. Number of Edges	30.77	32.30	32.13	62.14



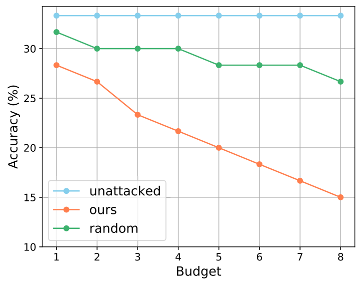
(a) NCI1



(b) NCI109



(c) Mutagenicity



(d) ENZYMES

Fig. 3: The impact of attack budget on unsupervised task performance.

Hyperparameter Setting. We use the default hyperparameters for the baselines. For the attack budget, we evaluate values of 1, 2, and 3, which represent the number of edges added or removed during the attack. Other hyper-parameters include a learning rate of 0.001, 100 epochs, and a batch size of 256.

5.2 Attack Evaluation

Attacking Unsupervised Models. We first conduct attack tests on unsupervised graph node embeddings. Given the unavailability of label information in unsupervised attack scenarios, we benchmark against random attacks due to the impracticality of conventional methods. Figure 3a, 3b, 3c and 3d illustrate the impact of our method on graph contrastive learning models under different attack budgets. With increasing attack budgets, the random method exhibits minimal overall variation. This is attributed to the fact that for graphs with

Table 2: Performance on GAT model.

Dataset	Method	k=1	k=2	k=3
NCI1	Unattacked	75.67	75.67	75.67
	Random	75.18	74.45	74.21
	Projective Ranking	73.72	70.31	68.61
	GradArgmax	74.93	74.69	73.96
	Ours	73.24	71.53	69.83
NCI109	Unattacked	71.19	71.19	71.19
	Random	71.19	70.94	70.46
	Projective Ranking	69.97	66.82	65.13
	GradArgmax	70.70	70.21	69.73
	Ours	69.73	68.04	66.34
Mutagenicity	Unattacked	74.42	74.42	74.42
	Random	73.96	73.50	72.35
	Projective Ranking	73.50	71.88	70.96
	GradArgmax	73.73	73.04	72.12
	Ours	73.27	71.43	70.51
ENZYMES	Unattacked	35.00	35.00	35.00
	Random	35.00	35.00	33.33
	Projective Ranking	30.00	26.67	23.33
	GradArgmax	35.00	33.33	33.33
	Ours	33.33	31.67	28.33

Table 3: Performance on GCN model.

Dataset	Method	k=1	k=2	k=3
NCI1	Unattacked	73.97	73.97	73.97
	Random	73.48	73.23	72.51
	Projective Ranking	71.28	68.61	66.18
	GradArgmax	71.77	70.31	68.85
	Ours	72.26	70.80	68.13
NCI109	Unattacked	69.98	69.98	69.98
	Random	69.25	68.52	67.07
	Projective Ranking	67.31	64.89	62.95
	GradArgmax	68.52	67.79	66.10
	Ours	68.77	67.31	64.65
Mutagenicity	Unattacked	72.81	72.81	72.81
	Random	72.35	72.12	71.89
	Projective Ranking	69.81	68.20	66.35
	GradArgmax	70.96	70.27	70.04
	Ours	71.20	70.05	69.59
ENZYMES	Unattacked	35.00	35.00	35.00
	Random	35.00	33.33	33.33
	Projective Ranking	26.67	23.33	20.00
	GradArgmax	33.33	33.33	31.67
	Ours	28.33	26.67	23.33

dozens or more edges, random edge modifications do not yield significant effects. In contrast, our method identifies edges that significantly influence model predictions, leading to improved attack performance.

Attacking supervised Models. Tables 2 and 3 provide a detailed comparison of our attack method against other techniques in supervised tasks on GCN model and GAT model. It is observed that our approach significantly outperforms the random method. Compared to GradArgmax, our approach consistently outperforms GradArgmax in attacks on the GAT model across multiple datasets and varying attack budgets. In attacks on the GCN model, although GradArgmax performs well on the NCI1, NCI109, and Mutagenicity datasets under lower attack budgets, as the attack budget increases, our method gradually surpasses it, demonstrating superior robustness. Compared to Projective Ranking, while Projective Ranking slightly outperforms our method in some cases, it does so at the cost of requiring access to the model's predictive probability distribution. This reliance on additional model-specific information can limit its practical applicability, especially in scenarios where such data is inaccessible. In contrast, our method operates independently of these constraints, offering a more universally applicable and streamlined attack strategy.

6 Conclusion

In this work, we introduce an unsupervised method utilizing graph contrastive learning to perform adversarial attacks in graph classification tasks in the absence of graph labels. Our experiments across diverse datasets, such as NCI1, NCI109, Mutagenicity and ENZYMES, show the effectiveness of our method in decreasing graph classification accuracy.

Acknowledgments. Zhiwei Zhang is supported by the National Key R&D Program of China (Grant No.2024YFE0209000), the NSFC (Grant No.U23B2019, No.U23B200400).

References

1. Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., Song, L.: Adversarial attack on graph structured data. In: ICML. pp. 1115–1124 (2018)
2. Jin, W., Barzilay, R., Jaakkola, T.: Junction tree variational autoencoder for molecular graph generation. In: ICML. pp. 2323–2332 (2018)
3. Lee, J.B., Rossi, R., Kong, X.: Graph classification using structural attention. In: SIGKDD. pp. 1666–1674 (2018)
4. Ma, Y., Wang, S., Derr, T., Wu, L., Tang, J.: Graph adversarial attack via rewiring. In: SIGKDD. pp. 1161–1169 (2021)
5. Mu, J., Wang, B., Li, Q., Sun, K., Xu, M., Liu, Z.: A hard label black-box adversarial attack against graph neural networks. In: SIGSAC. pp. 108–125 (2021)
6. Qiu, J., Chen, Q., Dong, Y., Zhang, J., Yang, H., Ding, M., Wang, K., Tang, J.: Gcc: Graph contrastive coding for graph neural network pre-training. In: SIGKDD. pp. 1150–1160 (2020)
7. Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., Tang, J.: Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In: WSDM. pp. 459–467 (2018)
8. Stokes, J.M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N.M., MacNair, C.R., French, S., Carfrae, L.A., Bloom-Ackermann, Z., et al.: A deep learning approach to antibiotic discovery. Cell **180**(4), 688–702 (2020)
9. Sun, L., Dou, Y., Yang, C., Zhang, K., Wang, J., Yu, P.S., He, L., Li, B.: Adversarial attack and defense on graph data: A survey. TKDE **35**(8), 7693–7711 (2023)
10. Xiong, Z., Wang, D., Liu, X., Zhong, F., Wan, X., Li, X., Li, Z., Luo, X., Chen, K., Jiang, H., et al.: Pushing the boundaries of molecular representation for drug discovery with the graph attention mechanism. Journal of medicinal chemistry **63**(16), 8749–8760 (2019)
11. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? In: ICLR (2019)
12. You, X., Wang, C.X., Huang, J., Gao, X., Zhang, Z., Wang, M., Huang, Y., Zhang, C., Jiang, Y., Wang, J., et al.: Towards 6g wireless communication networks: Vision, enabling technologies, and new paradigm shifts. SCIS **64**, 1–74 (2021)
13. You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., Shen, Y.: Graph contrastive learning with augmentations. NeurIPS **33**, 5812–5823 (2020)
14. Zhang, A., Ma, J.: Defensevgae: Defending against adversarial attacks on graph data via a variational graph autoencoder. In: ICIC. pp. 313–324 (2024)
15. Zhang, H., Yuan, X., Zhou, C., Pan, S.: Projective ranking-based gnn evasion attacks. TKDE **35**(8), 8402–8416 (2023)
16. Zhang, H., Zheng, T., Gao, J., Miao, C., Su, L., Li, Y., Ren, K.: Data poisoning attack against knowledge graph embedding. In: IJCAI. pp. 4853–4859 (2019)
17. Zhang, M., Cui, Z., Neumann, M., Chen, Y.: An end-to-end deep learning architecture for graph classification. AAAI **32**(1) (2018)