Projective Ranking: A Transferable Evasion Attack Method on Graph Neural Networks

He Zhang[1], Bang Wu[1], Xiangwen Yang[1], Chuan Zhou[2], Shuo Wang[3], Xingliang Yuan[1], Shirui Pan[1]
{he.zhang1,bang.wu,wayne.yang,xingliang.yuan,shirui.pan}@monash.edu
zhouchuan@amss.ac.cn,shuo.wang@data61.csiro.au
[1]Monash University, Australia [2]AMSS, Chinese Academy of Sciences, China [3]CSIRO, Australia

ABSTRACT

Graph Neural Networks (GNNs) have emerged as a series of effective learning methods for graph-related tasks. However, GNNs are shown vulnerable to adversarial attacks, where attackers can fool GNNs into making wrong predictions on adversarial samples with well-designed perturbations. Specifically, we observe that the current evasion attacks suffer from two limitations: (1) the attack strategy based on the reinforcement learning method might not be transferable when the attack budget changes; (2) the greedy mechanism in the vanilla gradient-based method ignores the long-term benefits of each perturbation operation. In this paper, we propose a new attack method named projective ranking to overcome the above limitations. Our idea is to learn a powerful attack strategy considering the long-term benefits of perturbations, then adjust it as little as possible to generate adversarial samples under different budgets. We further employ mutual information to measure the long-term benefits of each perturbation and rank them accordingly, so the learned attack strategy has better attack performance. Our method dramatically reduces the adaptation cost of learning a new attack strategy by projecting the attack strategy when the attack budget changes. Our preliminary evaluation results in synthesized and real-world datasets demonstrate that our method owns powerful attack performance and effective transferability.

CCS CONCEPTS

• **Security and privacy → Software and application security**;
• **Computing methodologies → Artificial intelligence**.

KEYWORDS

Graph neural networks, Graph classification, Adversarial attacks

ACM Reference Format:
He Zhang[1], Bang Wu[1], Xiangwen Yang[1], Chuan Zhou[2], Shuo Wang[3], Xingliang Yuan[1], Shirui Pan[1]. 2021. Projective Ranking: A Transferable Evasion Attack Method on Graph Neural Networks. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3459637.3482161

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482161

1 INTRODUCTION

A graph is an abstract data type with powerful modelling capabilities. It can be obtained by extracting attributes of entities and relations between them and then explored by graph learning methods in various real-world applications, such as social networks, commercial recommendation systems, biological networks, and traffic congestion prediction [14]. Graph neural networks (GNNs) have emerged as effective graph learning methods [7] and demonstrate promising performance on node classification [3, 13], link prediction [16, 21], and graph classification [22].

While GNNs have attracted considerable attention in various applications with complex graph-structured data [6], they also raise urgent security concerns in practice. Recent studies have shown that GNNs are vulnerable to adversarial attacks. Using the evasion attack as a representative example, attackers propose generating adversarial samples to fool the victim model by adding unnoticeable permutations on clean samples. The victim model will give wrong outputs on the adversarial samples, which are different from that of clean samples [1]. The evasion attack is notoriously dangerous for several reasons. First, well-designed slight perturbations will significantly degrade the performance of the victim model. Additionally, once the model is deployed, the attacker can launch evasion attacks at any time, which increases the difficulty and cost of mitigating them. Finally, this kind of attack is sneaky since the adversarial samples are similar to the clean samples. To obtain an adversarial example, attackers may perturb a clean graph by operations like adding/deleting few edges or modifying the features of some nodes. To make evasion attacks stealthy, attackers use a specified *budget* to restrict the number of these operations.

The current studies of evasion attacks on the GNN models mainly focus on node classification and link prediction, while few are designed for graph classification [4]. In current evasion attack methods for graph classification, two *limitations* remain to be solved to improve the attack practicality. First, some attack methods using reinforcement learning are not effective enough to generate powerful adversarial samples, especially when the budget is more than one [10]. When the budget changes, they also need to be retrained to learn an attack strategy suitable for the new budget. The adversary is eager for a powerful attack method suitable for various perturbation budgets and hopes to use existing attack strategies to reduce adjustment efforts when the perturbation budget changes. Second, the gradient-based method *GradArgmax* [1] is inefficient since it needs real-time gradient information to complete each step of the perturbation operations. In *GradArgmax*, the greedy mechanism based on the "local" gradient also neglects the "long-term" benefits of operation at each step, which is inconsistent with the adversary's desire to gain insight into the long-term benefits of each operation.

Motivated by these observations, we summarize the challenges in building an effective attack against GNNs for graph classification: (1) *how to transfer learned attack strategy to changed perturbation budget with least effort*, and (2) *how to measure long-term benefits of each operation in the generation of powerful adversarial samples*.

To address these challenges, we expect to find a suitable metric that considers the long-term benefits of each operation to rank all possible operations and then use this ranking directly to generate adversarial samples under different perturbation budgets. Inspired by the projected gradient descent method, we project a learned attack strategy into practical attackers' perturbations with the specified budget for the first challenge. Through this projection operation, the adjustment cost of our learned attack strategy is almost zero. Towards the second challenge, we regard the space composed of all possible perturbation operations as the *perturbation space*. We employ the mutual information between elements of *perturbation space* and attackers' goals as the measure to evaluate the importance of each possible operation. We regard the perturbation space ranking based on mutual information as the learned attack strategy.

Our contributions are summarized below:

- We propose the projection operation to reduce the cost of the strategy adaptation process by considering the transferability of the attack strategy when the budget changes.
- We employ mutual information to measure the importance of a perturbation operation towards strong attack performance.
- We conduct preliminary evaluations on a synthesized dataset and five real-world datasets, and our method owns powerful attack performance and effective transferability.

2 RELATED WORK

According to the intention of attackers, current adversarial attack methods mainly include evasion attacks, backdoor attacks, and poisoning attacks. Here we provide a short review of these attacks.

Evasion attacks perturb the inputs of the victim GNNs during their inference. In evasion attacks on graph classification, the attack successful rate of *RL-S2V* outperforms *RandSampling* and *GradArgmax* on synthetic data [1]. To improve the stealthiness of attacks, Ma et al. proposed *ReWatt* to redefine the action space of reinforcement learning [10]. In graph classification, there are mainly two different manners to obtain the expression of a graph. One way is performing global pooling on expressions of all nodes, such as the *mean-pooling* or *max-pooling*. Another one is introducing hierarchical pooling in the GNN models. The pooling operation selects typical nodes to compress the node number and determine the structure of the coarse graph [9, 18, 19]. Tang et al. proposed to attack the selecting operation in pooling operation [12], but it may not be used for attacking models that employ global pooling.

Another two typical adversarial attacks on GNN models are poisoning attacks and backdoor attacks. The poisoning attacks [2] suppose the adversary owns the ability to poison the training data of the victim models. In this way, the performance of the GNN models trained on poisoned data will degrade dramatically on clean samples. In backdoor attacks [15, 20], the adversary generally injects a fixed or adaptive trigger into the clean training data and changes their labels to the desired labels. As a result, the models trained on these data perform well on clean samples but predict the desired labels once the trigger is injected into the clean samples.

3 PROPOSED APPROACH

3.1 Attack Setting and Problem Formulation

Our method is a white-box attack method. Attackers are assumed to have access to the node embedding and predictive probability distribution of the victim model to obtain a *powerful* (i.e., high attack success rate) and *transferable* (i.e., available to changed perturbation budget) attack strategy.

Assuming a GNN model f_θ is employed to predict the category of graph data $\mathcal{D} = \{G_j\}_1^M$. In evasion attacks, an attacker $\mathcal{T}$ attempts to make unnoticeable perturbations on the original G to degrade the performance of this victim model f_θ. We use $\mathcal{T}_f(G)$ to indicate the attacker's perturbation on G which is specifically designed for the classifier f_θ. More precisely, the objective of the attacker is

$$\begin{aligned} \max \quad & \textstyle\sum_{j=1}^{M} \mathbb{I}\left(f_\theta(\widehat{G_j}) \neq y_j\right) \\ \text{s.t.} \quad & \widehat{G} = \mathcal{T}_f(G) \\ & \mathcal{I}(\widehat{G}, G) = 1. \end{aligned} \tag{1}$$

Here $\widehat{G}$ is the adversarial sample generated from the clean sample G, and $\mathbb{I}(\cdot)$ is an indicator function. $\mathcal{I}(\cdot,\cdot)$ is a similarity measure function whose output is 1 when two input samples are semantically the same and 0 otherwise.

3.2 Perturbation Space

An attacker can perturb a clean sample G to obtain an adversarial sample $\widehat{G}$. We formulate the process of this perturbation on graph G as

$$\widehat{G} = \mathcal{T}_f(G) = G + \Delta G, \tag{2}$$

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

$$\begin{aligned} \{\widehat{A}, \widehat{X}\} &= \{A + \Delta A, X + \Delta X\}, \\ \Delta A &= m_A \odot \left[\mathbb{I}(add) \cdot (I_A - A) + \mathbb{I}(del) \cdot (-A)\right], \\ \Delta X &= m_X \odot (I_X - 2X), \end{aligned} \tag{3}$$

where $A \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ is adjacency matrix, $X \in \{0,1\}^{|\mathcal{V}|\times d}$ is binary node features matrix, $I_A = 1^{|\mathcal{V}|\times|\mathcal{V}|} - I_{|\mathcal{V}|}$, $I_X = 1^{|\mathcal{V}|\times d}$, $|\mathcal{V}|$ is the number of nodes in G, and d is the size of node features. $m_A \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$ and $m_X \in \{0,1\}^{|\mathcal{V}|\times d}$ represent the masks of graph structure and node features respectively. $\mathbb{I}(add)/\mathbb{I}(del)$ 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 then 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 (1) is refined as

$$\mathcal{I}(\widehat{G}, G) = \mathbb{I}\left(||m_A||_1 + ||m_X||_1 \leq k\right), \tag{4}$$

where $||\cdot||_1$ is the L1 Norm, k is the budget of perturbations.

The *perturbation space* $\mathbb{T}(G)$ is defined as the set of all possible perturbation graph $\Delta G = \{\Delta A, \Delta X\}$. Specially, we define the *perturbation space* with budget $k = 1$ as

$$\mathbb{T}_1(G) = \{\Delta G \,\big|\, ||m_A||_1 + ||m_X||_1 = 1\}. \tag{5}$$

We call $\Delta G \in \mathbb{T}_1(G)$ an *element* of *perturbation space* $\mathbb{T}_1(G)$.

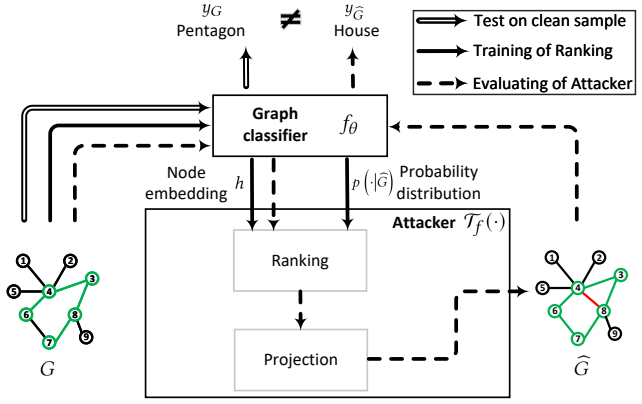

Figure 1: Illustration of the projective ranking method.

3.3 Projective Ranking

Now the attacker attempts to rank all elements in *perturbation space* $\mathbb{T}_1(G)$ and then use the ranking to generate an adversarial sample $\widehat{G}$. In Figure 1, the victim model f_θ can make the correct prediction "Pentagon" on G. During the ranking module training, the attacker $\mathcal{T}_f$ employs the mutual information between the perturbation space $\mathbb{T}_1(G)$ and the attacker's goal $y_{\widehat{G}} \neq y_G$ to measure and rank the importance of all elements in $\mathbb{T}_1(G)$. In the evaluation phase, the attacker $\mathcal{T}_f$ only needs to access the node embedding to obtain the ranking of the elements in $\mathbb{T}_1(G)$, and then projects the ranking to generate adversarial sample $\widehat{G}$ under the specified budget.

3.3.1 Ranking method. In this paper, the evasion attack strategy is embedded in the ranking of $\Delta G = \{\Delta A, \Delta X\} \in \mathbb{T}_1(G)$. Inspired by the interpretability study of GNNs [17], we measure the importance of elements in $\mathbb{T}_1(G)$ by mutual information **MI** between modified graph $\hat{G}$ and the attacker's goal. It can be expressed as:

$$\max \mathbf{MI}\left(\widehat{Y}, \widehat{G}\right) = H\left(\widehat{Y}\right) - H\left(\widehat{Y}|\widehat{G}\right), \tag{6}$$

where $H(\cdot)$ is entropy, $\widehat{Y} = \left(\cdots, p_{y_G} = 0, \cdots\right)$ is the expected prediction distribution of $\widehat{G}$, y_G is the original category of G. For a clean graph G, the first item in (6) is constant since the victim model f_θ is fixed and $\widehat{Y}$ is also invariant. So the objective (6) is equal to

$$\min H\left(\widehat{Y}|\widehat{G}\right). \tag{7}$$

To reduce the computational difficulties caused by the discrete graph structure, we apply continuous relaxation on ΔG and assume it is a graph variable $\Delta G \sim \mathcal{T}_f(G)$. Based on Jensen's inequality and $H(\cdot)$ is a concave function, we obtain

$$H\left(\widehat{Y}|\widehat{G}\right) = E_{\Delta G}\left[H\left(\hat{Y}|G + \Delta G\right)\right] \leq H\left(\widehat{Y}|G + E\left[\Delta G\right]\right), \tag{8}$$

where $E[\cdot]$ is the expectation function. So (7) is equal to

$$\min_{\mathcal{T}_f(G)} H\left(\widehat{Y}|G + E_{\mathcal{T}_f(G)}\left[\Delta G\right]\right). \tag{9}$$

To be more specific, given the victim model f_θ, the objective is

$$\min_{\mathcal{T}_f(G)} -\sum_y \mathbb{I}\left(y \neq y_G\right) \log p\left(y|\widehat{G}\right), \tag{10}$$

where $p\left(\cdot|\widehat{G}\right)$ is the predictive probability distribution of f_θ on $\widehat{G}$.

Algorithm 1 Generating Adversarial Samples

1: **Input**: clean samples $\mathcal{D} = \{G^m = \{A^m, X^m\}\}_1^M$, classifier f_θ
2: **Output**: adversarial samples $\widehat{\mathcal{D}} = \{\widehat{G^m} = \{\widehat{A^m}, X^m\}\}_1^M$
3: **Func:** $ranking\left(\mathcal{D}\right)$: return $\{\Delta A_c^m = (I_A - A) \odot s_\varphi(h_i^m, h_j^m)\}$
4: $h = NodeEmbedding\left(\mathcal{D}|f_\theta\right)$, $ASR_{best} = 0$, $k = 1$
5: **while** $NotEarlyStop$ **do**
6: // Attacker Training
7: $\{\Delta A_c^m\} = ranking\left(\mathcal{D}\right)$
8: $\{\widehat{G_c^m}\} = \{\{A^m + \Delta A_c^m, X^m\}\}$;
9: $\min_\varphi -\sum_{m=1}^M \sum_y \mathbb{I}\left(y \neq y_G\right) \log p\left(y|\widehat{G_c^m}\right)$
10: // Attacker Evaluation
11: $\{\Delta A_d^m\} = projection(ranking\left(\mathcal{D}\right))$
12: $\{\widehat{G_d^m}\} = \{\{A^m + \Delta A_d^m, X^m\}\}$;
13: **if** $ASR(\widehat{\mathcal{D}}|f_\theta) > ASR_{best}$ **then** $ASR_{best} = ASR_{current}$
14: **end while**

We assume the attacker is only able to add edges when generating $\widehat{G}$, which is common in the practical attack setting. So we have

$$\widehat{G} = \mathcal{T}_f(G) = \{A + \Delta A, X\}, \tag{11}$$

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

$$m_{Ai,j} = s\left(h_i, h_j\right) = softmax\left(MLP(concat(h_i, h_j))\right), \tag{12}$$

where h_i is the embedding of node i, $s(\cdot,\cdot)$ is the scoring function which measures the importance of elements in $\mathbb{T}_1(G)$ with respect to the attacker's expectation. Since the value of ΔA is a real number rather than a discrete value, we call $\widehat{G}$ is generated by adding continuous structure perturbations $\Delta G_c = \{\Delta A_c, 0\}$.

3.3.2 Perturbation Projection. Although the attacker obtains the adversarial sample $\widehat{G}$ in (11), it is necessary to bridge the gap between continuous perturbation ΔG_c and discrete modification ΔG_d under limited budgets. This is because the attacker can only operate one whole edge each time. Inspired by Projected Gradient Descent (PGD) algorithm, we map ΔG_c to ΔG_d by the projection function

$$\Delta A_{d;i,j} = \mathbb{I}\left(\Delta A_{c;i,j} \in topk\left(\Delta A_c, k\right)\right), \tag{13}$$

where k is the specified budget, $A_{c;i,j}$ is the element located at i-th row and j-th column of A_c, $topk(\cdot, k)$ is the set of $\Delta A_{c;i,j}$ with top-k values. After this projection, the attacker could fool the victim model f_θ by the adversarial sample $\widehat{G} = \{A + \Delta A_d, X\}$.

Algorithm 1 shows how an attacker generates adversarial samples from clean samples $\mathcal{D} = \{G^m = \{A^m, X^m\}\}_1^M$ by the proposed projective ranking method. In line 4, the attacker first needs to obtain the embedding representations h of all nodes in clean samples from the victim classifier f_θ. In the training phase (lines 6-9), the attacker uses the scoring function to obtain the structure mask m_A, and then ΔA_c by limiting m_A with the expected perturbation operation type. Next, the attacker uses ΔA_c to obtain $\widehat{G_c}$ (line 8) and trains the scoring function to learn the attack strategy (line 9). Then, the attacker projects the learned ranking (line 11, $k = 1$) to generate the adversarial sample $\widehat{G_d}$ (line 12). Finally, the attacker uses these adversarial samples to attack the victim model f_θ (line 13). The attacker will repeat the above process (lines 6-13) until the projective ranking method learns a powerful attack strategy.

Table 1: Classification Accuracy of the Victim GCN Model (%).

Dataset	BA-2Motifs			ENZYMES			Mutagenicity		
k	1	2	3	1	2	3	1	2	3
Clean	99.63	99.63	99.63	69.17	69.17	69.17	85.25	85.25	85.25
RandomSampling	78.85	54.35	50.09	64.81	56.33	51.23	80.80	75.56	70.75
GradArgmax	60.50	51.13	50.13	64.79	*61.46*	*58.13*	68.13	**54.50**	**45.50**
RL-S2V	**50.00**	**50.00**	**50.00**	58.54	56.04	*55.42*	71.88	69.50	63.75
Ours	**50.00**	**50.00**	**50.00**	**52.50**	**44.79**	**42.92**	**67.63**	59.13	53.63
Dataset	PC-3			NCI109			NCI-H23H		
k	1	2	3	1	2	3	1	2	3
Clean	67.13	67.13	67.13	76.50	76.50	76.50	64.75	64.75	64.75
RandomSampling	64.60	60.10	57.64	73.83	67.60	63.29	60.06	55.26	53.51
GradArgmax	*66.13*	*65.38*	*65.25*	*77.00*	*74.00*	*71.00*	56.13	53.25	52.63
RL-S2V	57.88	55.88	55.00	**66.00**	65.00	*64.13*	54.38	52.88	52.00
Ours	**56.13**	**52.13**	**49.62**	68.38	**59.00**	**56.75**	**53.13**	**50.25**	**50.13**

4 EXPERIMENTS

Datasets We employ one synthesized dataset BA-2Motifs [8] and five real-world datasets ENZYMES, Mutagenicity, PC-3, NCI109, NCI-H23H [11] for graph classification.

Baselines To evaluate the performance of our method, we select *RandomSampling*, *GradArgmax*, and *RL-S2V* [1] as the baselines.

Experimental Setup We randomly split each dataset into training data (80%), validation data (10%), and test data (10%) to train the victim GCN [5] model f_θ. The model has 3 hidden layers with 20 as the output feature dimension. We concatenate the *maxpooling* and *sumpooling* results of the final node embedding to obtain the graph expression, then use a fully connected layer to predict the category of the graph. In $T_f(\cdot)$, we use a 2-layers MLP to serve as the scoring module. To evaluate the *transferability* of our method, we directly project the learned ranking under $k = 1$ to generate adversarial samples under budget $k = 2, 3$. The results of other methods are obtained by running under a specified budget respectively.

Attack Performance and Transferability Table 1 shows the classification accuracy of the victim model f_θ on both clean samples and adversarial samples generated by different evasion attack methods. The bold numbers indicate the best attack results.

In the table1, our method achieves the best or competitive attack performance on almost all datasets with $k = 1$, indicating the projective ranking method owns *powerful* attack performance. When $k = 2$ or 3, it still holds the best or comparable attack effect, which shows that our method has learned *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 in each step. This greedy mechanism is not effective enough since ignoring the attacker's long-term gain. Our observation in table 1 ($k = 1$) shows that it performs worse on some datasets than *RL-S2V*, which considers the long-term gain in the evasion attack.

We also use *RandomSampling* as a baseline to evaluate other attack methods. Table1 shows our method always achieves better attack performance than *RandomSampling*. The *italic* accuracy show unexpected results in the aspect of attack performance. For *RL-S2V*, its attack performance is worse than *RandomSampling* on ENZYMES ($k = 3$) and NCI109 ($k = 3$) datasets. This may be due to the less effectiveness of the Q-learning method in *RL-S2V*, especially for the Markov decision process with long-horizon. For *GradArgmax*, its attack performance is worse than *RandomSampling* on ENZYMES ($k = 2, 3$), PC-3 ($k = 1, 2, 3$) and NCI109 ($k = 1, 2, 3$) datasets. Particularly, its adversarial samples own negative performance on NCI109 ($k = 1$). Maybe the local gradient information is not a suitable enough measure of attack benefits for generating adversarial samples.

5 CONCLUSION

In this paper, we propose the projective ranking method to make an evasion attack for graph classification. We argue that existing evasion methods either do not have sufficient attack performance due to ignoring the long-term benefits of the perturbations or require the attacker to adjust the attack strategy when the attack environment changes. To this end, we formulate the problem and the perturbation space for the evasion attack on graph classification. We first relax perturbation space to rank its elements based on the mutual information, then project the ranking in a practical attack with the specified budget. Experimental results show that our method exhibits strong attack performance, and the learned attack knowledge can be used to generate adversarial samples when the perturbation budget changes.

6 ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (No.61872360).

REFERENCES

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

[2] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. 2018. Poisoning Attacks to Graph-Based Recommender Systems. In *Proceedings of the 34th Annual Computer Security Applications Conference, 2018*. ACM, 381–392. https://doi.org/10.1145/3274694.3274706

[3] Ming Jin, Yizhen Zheng, Yuan-Fang Li, Chen Gong, Chuan Zhou, and Shirui Pan. 2021. Multi-Scale Contrastive Siamese Networks for Self-Supervised Graph Representation Learning. *arXiv preprint arXiv:2105.05682* (2021).

[4] Wei Jin, Yaxin Li, Han Xu, Yiqi Wang, and Jiliang Tang. 2020. Adversarial attacks and defenses on graphs: A review and empirical study. *arXiv preprint arXiv:2003.00653* (2020).

[5] Thomas N Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *International Conference on Learning Representations, ICLR* (2017).

[6] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. 2020. Distance Encoding: Design Provably More Powerful Neural Networks for Graph Representation Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems, 2020*.

[7] Yixin Liu, Shirui Pan, Ming Jin, Chuan Zhou, Feng Xia, and Philip S Yu. 2021. Graph self-supervised learning: A survey. *arXiv preprint arXiv:2103.00111* (2021).

[8] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. 2020. Parameterized Explainer for Graph Neural Network. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*.

[9] Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. 2019. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 723–731.

[10] Yao Ma, Suhang Wang, Lingfei Wu, and Jiliang Tang. 2019. Attacking Graph Convolutional Networks via Rewiring. *CoRR* abs/1906.03750 (2019). arXiv:1906.03750 http://arxiv.org/abs/1906.03750

[11] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*. arXiv:2007.08663 www.graphlearning.io

[12] Haoteng Tang, Guixiang Ma, Yurong Chen, Lei Guo, Wei Wang, Bo Zeng, and Liang Zhan. 2020. Adversarial Attack on Hierarchical Graph Pooling Neural Networks. *CoRR* abs/2005.11560 (2020). arXiv:2005.11560 https://arxiv.org/abs/2005.11560

[13] Man Wu, Shirui Pan, and Xingquan Zhu. 2020. OpenWGL: Open-World Graph Learning. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 681–690.

[14] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 753–763.

[15] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. 2021. Graph backdoor. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*.

[16] Bo Xiong, Shichao Zhu, Nico Potyka, Shirui Pan, Chuan Zhou, and Steffen Staab. 2021. Semi-Riemannian Graph Convolutional Networks. *arXiv preprint arXiv:2106.03134* (2021).

[17] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*. 9240–9251.

[18] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*. 4805–4815.

[19] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. 2019. Hierarchical Graph Pooling with Structure Learning. *CoRR* abs/1911.05954 (2019). arXiv:1911.05954 http://arxiv.org/abs/1911.05954

[20] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2020. Backdoor Attacks to Graph Neural Networks. *CoRR* abs/2006.11165 (2020). arXiv:2006.11165 https://arxiv.org/abs/2006.11165

[21] Shichao Zhu, Shirui Pan, Chuan Zhou, Jia Wu, Yanan Cao, and Bin Wang. 2020. Graph Geometry Interaction Learning. *Advances in Neural Information Processing Systems* 33 (2020).

[22] Shichao Zhu, Lewei Zhou, Shirui Pan, Chuan Zhou, Guiying Yan, and Bin Wang. 2020. Gssnn: graph smoothing splines neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 7007–7014.