Towards Anomaly-resistant Graph Neural Networks via Reinforcement Learning

Kaize Ding
Arizona State University
kaize.ding@asu.edu

Xuan Shan
Kwai Inc.
shanxuan@kuaishou.com

Huan Liu
Arizona State University
huan.liu@asu.edu

ABSTRACT

In general, graph neural networks (GNNs) adopt the message-passing scheme to capture the information of a node (i.e., nodal attributes, and local graph structure) by iteratively transforming, aggregating the features of its neighbors. Nonetheless, recent studies show that the performance of GNNs can be easily hampered by the existence of abnormal or malicious nodes due to the vulnerability of neighborhood aggregation. Thus it is necessary to learn anomaly-resistant GNNs without the prior knowledge of ground-truth anomalies, given the fact that labeling anomalies is costly and requires intensive domain knowledge. In order to keep the effectiveness of GNNs on anomaly-contaminated graphs, in this paper, we propose a new framework named RARE-GNN (Reinforced Anomaly-REsistant Graph Neural Networks) which can detect anomalies from the input graph and learn anomaly-resistant GNNs simultaneously. Extensive experiments on real-world datasets demonstrate the effectiveness of the proposed framework.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**;

KEYWORDS

Graph neural networks, Robustness, Reinforcement learning

ACM Reference Format:
Kaize Ding, Xuan Shan, and Huan Liu. 2021. Towards Anomaly-resistant Graph Neural Networks via Reinforcement Learning. 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/10.1145/3459637.3482203

1 INTRODUCTION

Real-world graphs are commonly contaminated with a small portion of nodes, namely, anomalies, whose patterns significantly deviate from the majority nodes [1, 2, 5, 6]. For instance, in a social network, there may exist camouflaged users who randomly follow different users, rendering properties like homophily not applicable to this type of relationships [10, 25]. Owing to the vulnerability of neighborhood aggregation according to previous research [9, 10, 27], the existence of such abnormal instances will inevitably deteriorate the performance of GNNs – unwanted messages from those abnormal nodes will be propagated throughout the graph, learning to the learned node representations less expressive. Therefore, how to learn anomaly-resistant graph neural networks is a challenging yet imperative problem to further push forward the performance boundary of existing works.

However, solving the aforementioned problem remains a non-trivial task. Since collecting ground-truth labels of anomalies is extremely expensive and requires intensive domain-knowledge [2, 6, 18], it is impractical to annotate substantial ground-truth anomalies. Upon the success of research on graph-based anomaly detection [2, 5, 6, 35], one natural solution is to apply one of the unsupervised anomaly detection methods on the input graph, and train the GNN model on a cleaned graph in a either joint or pre-post way. Although being intuitive, those two separate phases target disjoint optimization objectives, which in turn poses great challenge to systematically learn anomaly-resistant GNNs. Specifically, due to the lack of supervision, the anomaly detection phase may introduce severe learning errors: on the one hand, if normal nodes are wrongly detected as anomalies (false positive), it may cause information loss when learning GNNs on the target task; on the other hand, those undetected anomalies (false negative) will still hamper the model performance of learned GNNs. Hence, how to align the anomaly detection strategy with the final GNN model performance on the target task is another challenge to resolve.

To address the challenges discussed above, we propose a new GNN framework – Reinforced Anomaly-REsistant Graph Neural Networks (RARE-GNN) in this study. In essence, the proposed framework RARE-GNN is composed of two main graph neural modules: the *detection network* that detects and removes anomalies from the input graph and the *prediction network* that learns and evaluates on the "cleaned" graph for a down-stream task. Specifically, we model the studied problem as a Markov Decision Process (MDP) task and employ deep Q-learning (DQN) [20] as the medium to bridge the non-differentiable gap between the two graph neural modules. At each timestamp, a suspicious node will be selected by the *detection network* and the relations with its neighboring nodes will be updated accordingly. After a series of selections, an augmented graph [36] that masks out all the selected nodes and related edges will be fed to the *detection network*. Meanwhile, its selection policy will be evaluated by a reward derived from the *prediction network* [29, 32] on the target task. In this way, the two modules can seamlessly work together and those anomalies hamper the model performance on the target task can be accurately detected. We conduct extensive experiments on various datasets and the experimental results demonstrate the superior performance of RARE-GNN on node classification as well as anomaly detection over the state-of-the-arts.

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

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482203

2 RELATED WORK

Graph Representation Learning. Graphs serve as a common language for modeling relational and structured data. As the essential key for conducting various graph analytical tasks, how to learn expressive graph representations has drawn much research attention. Recently, graph neural networks (GNNs) have become a prevailing paradigm for graph representation learning due to its superior effectiveness [7, 12, 14, 15, 28, 30, 31]. In general, GNNs adopt a message-passing scheme and learn node representations by iteratively transforming and aggregating the information from local neighborhoods. Many models in this line of work such as GCN [14], GAT [28], and GraphSAGE [12], have achieved great success in both academic and industrial communities. However, the message-passing scheme relies on the homophily assumption [19] and is inherently vulnerable to outliers or anomalies [4, 10]. Existing GNN-based graph anomaly detection methods [4, 5, 35] commonly focus on detecting anomalies rather than learning anomaly-resistant GNNs. Though recent works like SEANO [17] and ONE [2] are able to learn robust representations that jointly preserve graph information while minimizing the negative effects of anomalies, none of the them is tailored for graph neural networks, rendering the learned node representations less expressive. As a necessary supplement in this research field, our RARE-GNN framework is compatible with arbitrary GNN architectures for any specific down-stream learning tasks. By integrating the anomaly detection process and GNN learning process into a unified framework, we are able to mitigate the adverse effects of anomalies and further learn powerful GNNs.

Reinforcement Learning on Graphs. Reinforcement learning (RL) offers a powerful approach to solve challenging problems in a variety of domains. More recently, reinforcement learning has begun to find applications that involve graph-structured data. As one pioneering work, Graph Convolutional Policy Network (GCPN) [33] uses RL to learn to generate molecular graphs. GraphUCB [6] extends contextual multi-armed bandit (MAB) to graph-structured data for detecting abnormal nodes. XGNN [34] generates graphs via RL to achieve model-level interpretation of GNN models. Do et al. [8] consider chemical reaction as markov decision process of graph transformation and propose to use RL for predicting the products of chemical reactions. As another line of application, people explore to perform adversarial attacks on graph-structured data using different RL agents [3, 26]. Recent works [16, 29] leverage RL to optimize the neighborhood aggregation functions for pushing forward the performance boundary of GNNs and Hu et al. [13] use RL to learn a transferable active learning policy which can directly generalize to unlabeled target graphs. Compared to the aforementioned methods, our work is the first attempt for learning anomaly-resistant GNNs by virtue of reinforcement learning.

3 PROPOSED APPROACH

3.1 Learning Environment

We start with introducing the learning environment of the problem. As shown in Figure 1, at each time step t, the *detection network* selects one action a_t (i.e., an abnormal node) based on the current state s_t that represents the current graph. Then the *prediction network* will return a reward r_t by quantifying the model performance

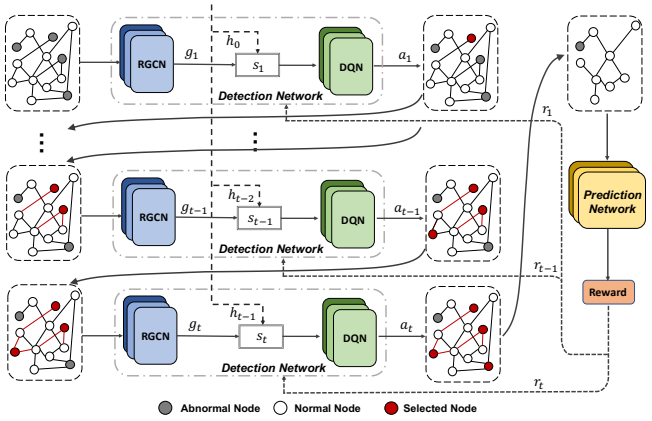


Figure 1: Overview of the proposed RARE-GNN framework. In each time step, the *detection network* selects one node, and the *prediction network* will be trained on the anomaly-removed graph to update the selection policy using the computed reward on the validation set.

on a clean validation set, and uses it as a reinforcement signal to learn the abnormality of each node being used in the training of the predictor model. According to the reward, the reinforced *detection network* updates its policy and the state will be changed to s_{t+1}. This process continues until the agent reaches a termination state. Intuitively, each training episode can be formulated as a Markov Decision Process (MDP) characterized by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, in which $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{T}$ is the transition function, $\mathcal{R}$ is the possible rewards, and γ is the discount factor. Mathematically, the objective is to learn a policy $\pi(\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ to maximize the accumulated discounted rewards:

$$\pi(\boldsymbol{\theta})^* = \max \sum_{t=0}^{\infty} \gamma^t r_t, \tag{1}$$

where r_t denotes the immediate reward at timestamp t. Specifically, in our problem, the agent only receives reward at the end. Specifically, the MDP tuple is defined as follows:

State $\mathcal{S}$. The state $s_t = [\mathbf{g}_t, \mathbf{h}_{t-1}]$ which encodes the information of the current graph G_t after previous selections, is represented by the concatenation of the hidden state of last step $\mathbf{h}_{t-1}$ and the intermediate graph representation $\mathbf{g}_t$.

Action $\mathcal{A}$. Given all the candidates nodes, the policy maps the state $\mathbf{s}_t$ into an action a_t at each time step t. Here a_t is either selecting a node in $\mathcal{V}$ or selecting the terminal action. Note that if a node has been selected in previous time steps, it will not be considered again in following iterations.

Transition $\mathcal{T}$. It represents the function of transition from s_t to s_{t+1}, where s_{t+1} is considered to be a possible result of selecting an action in s_t. In our case, the transition function $\mathcal{T}$ is deterministic, which means the next state s_{t+1} is not stochastic and only depends on the current state and action pair (s_t, a_t).

Reward $\mathcal{R}$. The reward $r_t \in \mathcal{R}$ indicates whether the performance of the *detection network* can be improved after termination, which can be used to guide the agent to update its policy. In this paper, we

define the reward according to the down-stream task performance (e.g., node classification, link prediction) on the clean validation set with removing the selected nodes. Note that the model only receives reward when the task terminates, and we assign a zero reward to intermediate steps.

3.2 Reinforced Anomaly-resistant GNN

Given the state representation s_t, RARE-GNN will select an action according to the policy that can can maximize the final reward (down-stream task performance). In this section, we first illustrate how we compute the state representation $s_t = [\mathbf{g}_t, \mathbf{h}_{t-1}]$.

Graph Representation Learning. We first introduce how we build the graph representation module. Essentially, this module encodes the input attributed graph to a low-dimensional embedding vector. As shown in Figure 1, once the agent selects one node v_t as an abnormal node at step t, we are able to update the intermediate graph from G_t to G_{t+1} by changing the relation type between v_t and its neighbors. Correspondingly, for each node v_i, it could have two types of neighboring nodes: (1) *true neighbors*, which are considered as normal nodes so far; and (2) *false neighbors*, which have been selected as anomalies in previous time steps. During the neighborhood aggregation process, we should separately considered them to learn more expressive network representations.

To this end, we propose to use a relational GNN model [22] to learn the graph representation $\mathbf{g}_t$. Specifically, given a specific node v_i in the attributed graph, we use the following operation to calculate the representation of node v_i at the l-th layer:

$$\mathbf{h}_i^l = \sum_{r \in \{r^+, r^-\}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{c_i^r} \mathbf{W}_r^l \mathbf{h}_j^{l-1} + \mathbf{W}_0^l \mathbf{h}_i^{l-1}, \tag{2}$$

where N_i^r denotes the neighbors of node v_i in terms of type r. c_i^r is a normalization constant that can either be learned or chosen in advance, here we set $c_i^r = |\mathcal{N}_i^r|$.

To further compute the representation of the entire graph G_t. Following previous research, we directly compute the graph representation by taking the average of L-th layer node representations:

$$\mathbf{g}_t = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \mathbf{h}_i^L, \tag{3}$$

where $\mathbf{g}_t$ denotes the graph representation of G_t. Note that other graph pooling operation can also be used in our approach.

As we mentioned before, once the agent select one node v_t as an abnormal node at step t, we are able to update the intermediate graph from G_t to G_{t+1} by changing the relation type between v_t and its neighbors.

Hidden State Representation Learning. Considering that the agent performs sequential decision making, we introduce a hidden vector $\mathbf{h}_t$ to keep the history information to better guide the agent. Specifically, the hidden state $\mathbf{h}_t$ is determined by the last hidden state $\mathbf{h}_{t-1}$ and the current state $\mathbf{s}_t$:

$$\mathbf{h}_t = \text{GRU}(\mathbf{s}_t, \mathbf{h}_{t-1}), \tag{4}$$

where GRU (Gated Recurrent Unit) is a gating mechanism to control the memorization or forgetting of the history information.

Table 1: Details of the real-world attributed graphs

Dataset	Cora	CiteSeer	PubMed	MS-CS
# nodes	2,485	2,110	19,717	18,333
# edges	5,069	3.668	44,324	81,894
# attributes	1,433	3,703	500	6,805
# labels	7	6	3	15
# anomalies	50	30	911	842

Prediction Network. The *prediction network* is designed to learn node representations on the augmented graph for a specific down-stream task, i.e., node classification. Specifically, it can be built with arbitrary GNN architectures and the downstream task performance on the validation set is used as the reward signal for updating the selection policy. In our implementation, we adopt a two-layer GAT as the backbone. For node classification, the prediction network is trained on the augmented graph with a cross-entropy objective:

$$\mathcal{L}_{cls} = -\sum_{c=1}^{C} y_c \log(\hat{y}_c), \tag{5}$$

where $\mathbf{y}$ is the ground-truth and $\hat{\mathbf{y}}$ is the predicted label vector.

Detection (Policy) Network. As the model performance in terms of Accuracy and the discrete action are not differentiable, to train the proposed framework, we adopt Deep Q-learning as the medium to bridge the gap between the selection strategy and the target GNN model performance. Specifically, we use the experience replay technique with memory buffer $\mathcal{M}$ and adopt a two-layer MLP to build the *detection network*. We simulate action selection and store the resulting data in a memory buffer $\mathcal{M}$. In addition, we use ϵ-greedy policy to control the exploration-exploitation trade-off of our framework. Our goal is to minimize the Q-learning loss function $\mathcal{L}_{rl}$ as follows:

$$\mathbb{E}_{(s_t, a_t, s_{t+1}) \sim \mathcal{M}}[(r_t + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}; \boldsymbol{\theta}^*) - Q(s_t, a_t; \boldsymbol{\theta}))^2], \tag{6}$$

where $Q(s_t, a_t; \boldsymbol{\theta})$ is the Q-function, which estimates the expected reward of action a_t at state s_t.

4 EXPERIMENTS

4.1 Experimental Settings

Evaluation Datasets. In order to make a fair and comprehensive evaluation, we adopt four widely used graph datasets: **Cora-ML**, **CiteSeer**, **PubMed** and **MS-CS** in our experiments. The first three datasets are citation graphs, where each node represents a paper and the edges represent citations between them, while the last one is built from Microsoft Academic Graph where nodes represent authors and edges represent the coauthorship relations between them. All these datasets are benchmark datasets used for evaluating semi-supervised node classification [14, 23]. The summary of dataset statistics is presented in Table 1. We follow the standard training/validation/test splits as in previous studies [14, 23]. Specifically, we use 20 nodes per class for training, 500 nodes for validation, and 1000 nodes for testing. The proposed approach and baselines are all trained and evaluated with the complete graph structure and node features in the training dataset, without using

Table 2: Semi-supervised node classification results (mean accuracy ± standard deviation) on four datasets.

Methods	Cora		Citeseer		PubMed		MS-CS	
	ACC_{ori}	ACC_{ptb}	ACC_{ori}	ACC_{ptb}	ACC_{ori}	ACC_{ptb}	ACC_{ori}	ACC_{ptb}
DeepWalk [21]	70.7 ± 0.6	68.9 ± 0.8	43.2 ± 0.5	41.6 ± 0.7	65.3 ± 0.6	63.3 ± 0.7	68.6 ± 0.9	67.3 ± 0.9
node2vec [11]	65.2 ± 1.1	63.4 ± 1.0	41.7 ± 0.8	40.1 ± 0.5	61.4 ± 0.9	59.8 ± 1.0	66.3 ± 0.7	64.2 ± 1.0
GCN [14]	81.4 ± 0.4	76.3 ± 0.9	70.9 ± 0.5	66.5 ± 0.6	79.0 ± 1.1	75.2 ± 0.7	91.3 ± 0.2	87.5 ± 0.4
GAT [12]	83.3 ± 0.7	77.2 ± 1.0	72.6 ± 0.6	67.6 ± 0.8	78.5 ± 0.3	74.6 ± 0.4	90.5 ± 0.6	86.6 ± 0.4
SGC [30]	81.0 ± 0.0	76.9 ± 0.5	71.9 ± 0.1	66.8 ± 0.3	78.9 ± 0.4	75.5 ± 0.7	91.0 ± 0.2	87.1 ± 0.6
SEANO [17]	82.0 ± 0.5	78.1 ± 0.6	**74.3 ± 0.6**	71.6 ± 0.8	79.7 ± 0.4	76.8 ± 0.5	87.2 ± 0.5	86.3 ± 0.5
ONE [2]	77.5 ± 0.9	75.9 ± 0.8	69.8 ± 0.6	67.3 ± 0.3	75.3 ± 0.9	73.6 ± 0.8	84.8 ± 0.6	82.5 ± 0.9
RARE-GNN	**83.7 ± 0.6**	**82.2 ± 0.4**	74.0 ± 0.4	**73.3 ± 0.6**	**79.9 ± 0.4**	**77.8 ± 0.5**	**91.4 ± 0.3**	**90.7 ± 0.5**

the node labels in the held-out validation and testing sets. The model hyper-parameters are selected based on the performance on the validation set and the final classification accuracy is reported on the test set.

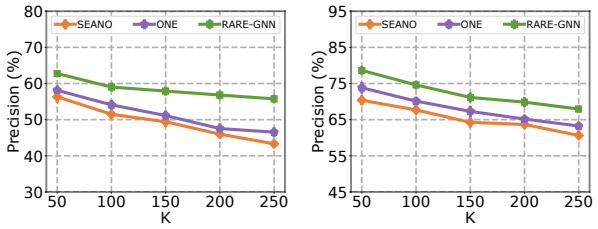
To evaluate robustness of different methods to anomalies, we inject a combined set of anomalies (i.e., structural anomalies and contextual anomalies) into each of the datasets. Specifically, we randomly select 5% of the nodes from a dataset (excluding validation and test set) and conduct perturbations to get the injected anomalies. For structural anomalies, we follow the method proposed by Ding et al. [5] to construct small cliques; for contextual anomalies, we follow the perturbation scheme described by Song et. al [17, 24] to modify the node attributes. Note that we inject structural and contextual anomalies with the same quantity.

Compared Methods. In our experiments, we compare the proposed RARE-GNN framework with three different categories of baseline methods: (1) *random walk-based* methods, including DeepWalk [21], node2vec [11]; (2) *GNN-based* models including GCN [14], GAT [28], SGC [30]; and (3) two state-of-the-art network embedding methods developed for anomaly-contaminated graphs. Specifically, SEANO [17] is a semi-supervised method to learn robust network embeddings while accounting for effects of anomalies, ONE [2] is an unsupervised attributed network embedding approach that jointly learns and minimizes the effect of anomalies in the network. It is worth pointing out that, for unsupervised methods (DeepWalk, node2vec and ONE), we train a MLP classifier based on the learned node representations to further conduct node classification. For a fair comparison, those baselines use the same training/validation/test data splits as other semi-supervised methods for training the MLP classifier.

4.2 Experimental Results

Node Classification Performance. As we are pursing more powerful GNNs that can eliminate the detrimental effect of anomalies, we first evaluate the model performance on one important graph learning task, i.e., semi-supervised node classification. Briefly, this task aims to predict the missing node labels with a small portion of labeled nodes. We repeat the evaluation process 10 times and report the average performance in terms of Accuracy in Table 2. Note that ACC_{ori} denotes the Accuracy on the original dataset, while ACC_{ptb} denotes the Accuracy on the perturbed (anomaly-injected) dataset. The following observations can be made from the table:

- For GNN-based methods including GCN, SGC and GAT, we can notice that there is a considerable gap between ACC_{ori} and ACC_{ptb}. It verifies that the GNN performance is very sensitive to the existence of anomalies. Compared to GNN-based methods, conventional graph embedding methods are not very sensitive to that, but still, become worse with injected anomalies.
- Overall, RARE-GNN is the best performing method on all the four datasets in terms of Accuracy (ACC). By using the down-stream task performance on the validataion set as the reinforcement signal, our graph anomaly detector is able to accurately detect those anomalies. In this way, the GNN predictor will be trained on a clean graph without those detected anomalies and the negative impact of anomalies will be alleviated.



(a) **PubMed** (b) **MS-CS**

Figure 2: Anomaly detection results (Precision@K) of different methods on PubMed and MS-CS.

Anomaly Detection Performance. In the experiments, we evaluate the performance of our proposed framework RARE-GNN by comparing it with other baseline methods. We present the experimental results in terms of Precision@K on the four datasets in Figure 2. Note that here we only include the results of SEANO and ONE as they are able to detect anomalies during the learning process. From the evaluation results, we see that the proposed deep framework RARE-GNN outperforms both SEANO and ONE by a noticeable margin on all the evaluation datasets. Especially, RARE-GNN can better rank anomalies on top positions. It shows that the *detection network* in RARE-GNN is able to accurately detect anomalies by learning the selection policy through RL.

REFERENCES

[1] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: a survey. *DMKD*, 2015.

[2] S. Bandyopadhyay, N. Lokesh, and M. N. Murty. Outlier aware network embedding for attributed networks. In *AAAI*, 2019.

[3] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song. Adversarial attack on graph structured data. In *ICML*, 2018.

[4] K. Ding, J. Li, N. Agarwal, and H. Liu. Inductive anomaly detection on attributed networks. In *IJCAI*, 2020.

[5] K. Ding, J. Li, R. Bhanushali, and H. Liu. Deep anomaly detection on attributed networks. In *SDM*, 2019.

[6] K. Ding, J. Li, and H. Liu. Interactive anomaly detection on attributed networks. In *WSDM*, 2019.

[7] K. Ding, J. Wang, J. Li, K. Shu, C. Liu, and H. Liu. Graph prototypical networks for few-shot learning on attributed networks. In *CIKM*, 2020.

[8] K. Do, T. Tran, and S. Venkatesh. Graph transformation policy network for chemical reaction prediction. In *KDD*, 2019.

[9] H. Dong, J. Chen, F. Feng, X. He, S. Bi, Z. Ding, and P. Cui. On the equivalence of decoupled graph convolution network and label propagation. In *The Web Conference*, 2021.

[10] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *CIKM*, 2020.

[11] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

[12] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[13] S. Hu, Z. Xiong, M. Qu, X. Yuan, M.-A. Côté, Z. Liu, and J. Tang. Graph policy network for transferable active learning on graphs. In *NeurIPS*, 2020.

[14] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[15] J. Klicpera, A. Bojchevski, and S. Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2019.

[16] K.-H. Lai, D. Zha, K. Zhou, and X. Hu. Policy-gnn: Aggregation optimization for graph neural networks. In *KDD*, 2020.

[17] J. Liang, P. Jacobs, J. Sun, and S. Parthasarathy. Semi-supervised embedding in attributed networks with outliers. In *SDM*, 2018.

[18] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis. Anomaly detection on attributed networks via contrastive self-supervised learning. *TNNLS*, 2021.

[19] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 2001.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

[21] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014.

[22] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

[23] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[24] X. Song, M. Wu, C. Jermaine, and S. Ranka. Conditional anomaly detection. *TKDE*, 2007.

[25] L. Sun, Y. Dou, C. Yang, J. Wang, P. S. Yu, L. He, and B. Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

[26] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar. Non-target-specific node injection attacks on graph neural networks: A hierarchical reinforcement learning approach. 2020.

[27] X. Tang, Y. Li, Y. Sun, H. Yao, P. Mitra, and S. Wang. Transferring robustness for graph neural network against poisoning attacks. In *WSDM*, 2020.

[28] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.

[29] L. Wang, W. Yu, W. Wang, W. Cheng, W. Zhang, H. Zha, X. He, and H. Chen. Learning robust representations with graph denoising policy network. In *ICDM*, 2019.

[30] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger. Simplifying graph convolutional networks. In *ICML*, 2019.

[31] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? 2019.

[32] J. Yoon, S. O. Arik, and T. Pfister. Data valuation using reinforcement learning. In *ICML*, 2020.

[33] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *NeurIPS*, 2018.

[34] H. Yuan, J. Tang, X. Hu, and S. Ji. Xgnn: Towards model-level explanations of graph neural networks. In *KDD*, 2020.

[35] T. Zhao, C. Deng, K. Yu, T. Jiang, D. Wang, and M. Jiang. Error-bounded graph anomaly loss for gnns. In *CIKM*, 2020.

[36] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah. Data augmentation for graph neural networks. In *AAAI*, 2021.