Adversarially Robust Neural Architecture Search for Graph Neural Networks

Beini Xie[1*] Heng Chang[1*] Ziwei Zhang[1] Xin Wang[1†] Daixin Wang[2]
Zhiqiang Zhang[2] Rex Ying[3] Wenwu Zhu[1†]
[1]Tsinghua University [2]Ant Group [3]Yale University

Abstract

Graph Neural Networks (GNNs) obtain tremendous success in modeling relational data. Still, they are prone to adversarial attacks, which are massive threats to applying GNNs to risk-sensitive domains. Existing defensive methods neither guarantee performance facing new data/tasks or adversarial attacks nor provide insights to understand GNN robustness from an architectural perspective. Neural Architecture Search (NAS) has the potential to solve this problem by automating GNN architecture designs. Nevertheless, current graph NAS approaches lack robust design and are vulnerable to adversarial attacks. To tackle these challenges, we propose a novel Robust Neural Architecture search framework for GNNs (G-RNA). Specifically, we design a robust search space for the message-passing mechanism by adding graph structure mask operations into the search space, which comprises various defensive operation candidates and allows us to search for defensive GNNs. Furthermore, we define a robustness metric to guide the search procedure, which helps to filter robust architectures. In this way, G-RNA helps understand GNN robustness from an architectural perspective and effectively searches for optimal adversarial robust GNNs. Extensive experimental results on benchmark datasets show that G-RNA significantly outperforms manually designed robust GNNs and vanilla graph NAS baselines by 12.1% to 23.4% under adversarial attacks.

1. Introduction

Graph Neural Networks are well-known for modeling relational data and are applied to various downstream real-world applications like recommender systems [37], knowledge graph completion [25], traffic forecasting [6], drug production [45], etc. Meanwhile, like many other deep neural networks, GNNs are notorious for their vulnerability under adversarial attacks [33], especially in risk-sensitive domains, such as finance and healthcare. Since GNNs model node representations by aggregating the neighborhood information, an attacker could perform attacks by perturbing node features and manipulating relations among nodes [55]. For example, in the user-user interaction graph, a fraudster may deliberately interact with other important/fake users to mislead the recommender system or fool credit scoring models [36].

A series of defense methods on graph data have been developed to reduce the harm of adversarial attacks. Preprocessing-based approaches like GCN-SVD [8] and GCN-Jaccard [38] conduct structure cleaning before training GNNs, while attention-based models like RGCN [51] and GNN-Guard [46] learn to focus less on potential perturbed edges. However, these methods rely on prior knowledge of the attacker. For example, GCN-SVD leverages the high-rank tendency of graph structure after Nettack [53], and GCN-Jaccard depends on the homophily assumption on the graph structure. As a result, current approaches may fail to adapt to scenarios when encountering new data and tasks or when new attack methods are proposed. Additionally, previous methods largely neglect the role of GNN architectures in defending against adversarial attacks, lacking an architectural perspective in understanding GNN robustness.

In order to reduce human efforts in neural architecture designs, Neural Architecture Search (NAS) has become increasingly popular in both the research field and industry. Though NAS has the potential of automating robust GNN designs, existing graph NAS methods [1, 10, 24, 49, 50] are inevitably susceptible to adversarial attacks since they do not consider adversarial settings and lack robustness designs [48]. Therefore, how to adopt graph NAS to search for optimal robust GNNs in various environments, and in the meantime, fill the gap of understanding GNN robustness from an architectural perspective, remains a huge challenge.

To address the aforementioned problems and to understand GNN robustness from an architectural perspective, we propose a novel **R**obust **N**eural **A**rchitecture search framework for **G**raph neural networks (G-RNA), which is the first attempt to exploit powerful graph neural architecture search in robust GNNs, to our best knowledge. Specifically, we first design a novel, expressive, and robust search space with

[*] Equal contribution.
[†] Corresponding authors.

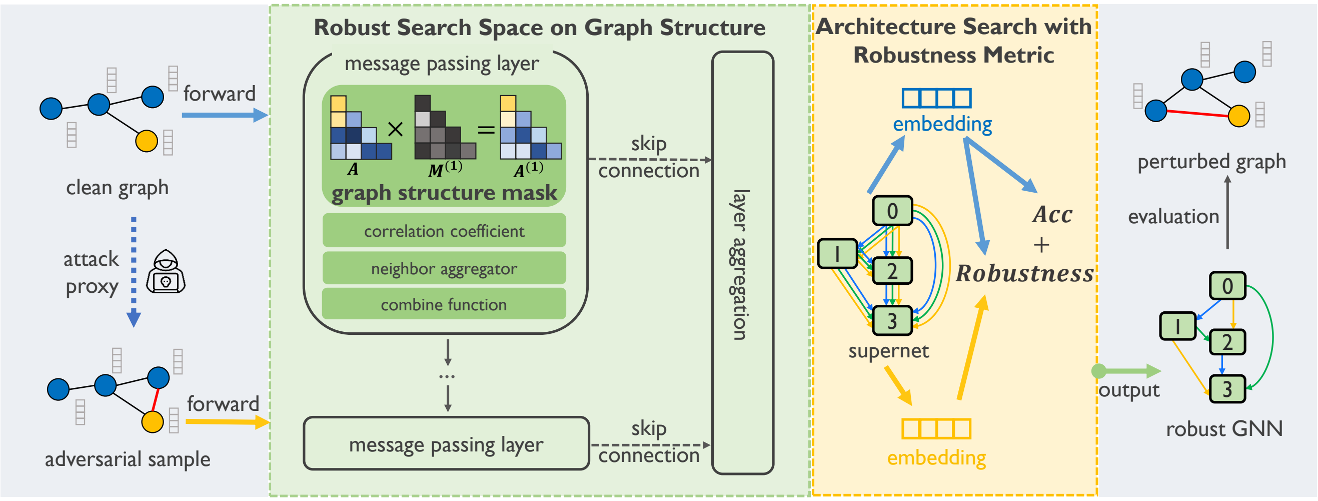


Figure 1. The overall framework of G-RNA. Given a clean graph, the supernet built upon our robust search space is trained in a single-path one-shot way. Then, the attack proxy produces several adversarial samples based on the clean graph and we search for robust GNNs with the proposed robustness metric. Finally, we evaluate the optimal robust GNN on graphs perturbed by the attacker.

graph structure mask operations. The green part in Fig. 1 shows the fine-grained search space. The graph structure mask operations cover important robust essences of graph structure and could recover various existing defense methods as well. We train the supernet built upon our designed search space in a single-path one-shot way [14]. Second, we propose a robustness metric that could properly measure the architecture's robustness. Based on the clean graph, an attack proxy produces several adversarial samples. We search robust GNNs using our robustness metric with clean and generated adversarial samples. A simple illustration of the robustness metric is shown in the yellow part in Fig. 1. After searching for the optimal robust GNN architecture with the evolutionary algorithm, we retrain the top-selected robust architectures from scratch and perform evaluations. Our contributions are summarized as follows:

- We develop a robust neural architecture search framework for GNNs, which considers robust designs in graph neural architecture search for the first time to the best of our knowledge. Based on this framework, we can understand adversarial robustness for GNNs from an architectural perspective.

- We propose a novel robust search space by designing defensive operation candidates to automatically select the most appropriate defensive strategies when confronting perturbed graphs. Besides, we design a robustness metric and adopt an evolutionary algorithm together with a single-path one-shot graph NAS framework to search for the optimal robust architectures.

- Extensive experimental results demonstrate the efficacy of our proposed method. G-RNA outperforms state-of-the-art robust GNNs by 12.1% to 23.4% on benchmark datasets under heavy poisoning attacks.

2. Related Work

2.1. Adversarial Robustness on Graph Data

Despite the wide success of GNNs in various applications [43, 44, 47], GNNs are shown vulnerable to adversarial attacks [2–4, 33, 39, 53], i.e., slight perturbations to graph can lead to sharp performance decrease. Following the literature, adversarial attacks have various taxonomies according to the attacker's knowledge (white-box attack and black-box attack), perturbation type (structure attack and feature attack), attack stage (evasion attack and poisoning attack), and targeted nodes (targeted attack and non-targeted attack).

In response to adversarial attacks, several defensive models have been proposed to enhance the robustness of GNNs. Graph pre-processing methods identify and rectify potential structural perturbations before the GNN model training. For example, with the assumption of feature smoothness, GCN-Jaccard [38] removes edges that have a low Jaccard similarity. Observing the high-rank tendency of the adjacency matrix under Nettack, GCN-SVD [8] reconstructs the adjacency matrix via its low-rank approximation. Graph attention methods aim to learn fewer attention weights on susceptible edges/features. For example, RGCN [51] uses Gaussian distribution for hidden layer node representations and calculates attention based on their variance. Pro-GNN [20] jointly learns graph structure and model parameters by keeping a low-rank and sparse adjacency matrix as well as feature smoothness. GNN-Guard [46] learns to focus more on edges between similar nodes and pruning edges between unrelated nodes. VPN [19] defenses by re-weighting edges from graph powering considering r-hop neighbors. Besides, graph certificate robustness methods [18, 35] provide a theoretical guarantee for graphs to be certified as robust under perturbation budgets. However, this branch is out of the scope of this work and we leave the comparison as a future work.

All previous studies rely heavily on manual designs and thus cannot adapt to new data, tasks, or adversarial attacks. Besides, existing methods neglect the inherent robustness of GNN architectures. On the contrary, our proposed method with specifically designed search space on graph structure can automatically search for the optimal robust GNN for different data and tasks.

2.2. Graph Neural Architecture Search

Neural Architecture Search (NAS) is a proliferate research direction that automatically searches for high-performance neural architectures and reduces the human efforts of manually-designed architectures. NAS on graph data is challenging because of the non-Euclidean graph property and special neural architectures [42, 48]. GraphNAS [10] uses the recurrent network as the controller to generate GNN architectures and adopts reinforcement learning to search for optimal architectures. In order to conduct an efficient search, differentiable NAS approaches [14, 26] jointly optimize the model weights and architecture parameters. DSS [24] proposes a differentiable one-shot graph NAS and dynamically updates the search space. SANE [49] also utilizes a differentiable search algorithm and builds GNN architectures with the Jumping Knowledge Network (JK-Net) backbone [41]. GNAS [1] reconstructs GNNs with the designed GNN Paradigm and learns the optimal message-passing depth as well. GASSO [30] uses graph structure learning as a denoising process in the differentiable architecture searching process. Graph NAS is also used in complex graph data such as heterogeneous graphs [7] and temporal graphs [29]. However, the existing graph NAS methods do not consider robustness against adversarial attacks.

2.3. Robust Neural Architecture Search

Robust neural architecture search exploits NAS to search for adversarially robust neural architectures. Since there is no related work for robust NAS on graph data, we review two remotely related papers on computer vision. RobNets [13] is the first work to explore architecture robustness. Through one-shot NAS, RobNets finetune architecture candidates via adversarial training and then sample more robust architectures. DSRNA [16] proposes two differentiable metrics which help to search robust architectures by a differentiable search algorithm. Our work is neither based on adversarial training nor adopts continuous relaxation for architecture parameters. To conclude, our work differs in that we consider a disparate search space tailored for graph data and leverage a different search algorithm.

3. Preliminaries

3.1. Graph Neural Networks

Let $\mathcal{G} = (\mathbf{A}, \mathbf{X})$ denote a graph with N nodes, where $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix and $\mathbf{X} \in \mathbb{R}^{N \times D_0}$ is the corresponding feature matrix. For node i, its neighborhood is denoted as $\mathcal{N}(i)$.

Graph Neural Networks take the graph data as input and output node/graph representations to perform downstream tasks like node classification and graph classification. Typically, for node classification tasks with C labels, we calculate:

$$\mathbf{z}_i = \left(f_{\boldsymbol{\alpha}}\left(\mathbf{A}, \mathbf{X}\right)\right)_i, \tag{1}$$

where $\mathbf{z}_i \in \mathbb{R}^C$ is the prediction vector for node i, $f_{\boldsymbol{\alpha}}$ denotes the graph neural network based on architecture $\boldsymbol{\alpha}$. The design of GNNs could be divided into intra-layer design, inter-layer design, and learning configurations [42]. Intra-layer design often follows the message-passing paradigm [11]: nodes representation is updated by aggregating neighborhood information. A general formula for updating node representation in GNNs is denoted as

$$\mathbf{h}_i^{(l)} = \sigma\left(\mathbf{W}^{(l)} \operatorname{Comb}\left(\mathbf{h}_i^{(l-1)}, \operatorname{Aggr}(e_{ij}^{(l)} \mathbf{h}_j^{(l-1)}, j \in \tilde{\mathcal{N}}(i))\right)\right), \tag{2}$$

where $\mathbf{h}_i^{(l)}$ denotes the node representation for node i in the l-th hidden layer, $e_{ij}^{(l)}$ is the correlation coefficient between node i and j, $\tilde{\mathcal{N}}(i) = \{i\} \cup \mathcal{N}(i)$ represents the neighborhood of node i with the self-loop, $\operatorname{Aggr}(\cdot)$ is the function to aggregate neighborhood information, $\operatorname{Comb}(\cdot)$ aims for combining self- and neighbor-information, and $\sigma(\cdot)$ is the activation function.

Besides the GNN layer design, how to connect different hidden layers is also critical. Some GNNs directly use the last hidden layer output as the prediction, while pre-processing layers, post-processing layers, and skip connections could also be added [22]. As for learning configurations, they are hyper-parameters for training GNNs like the learning rate, the optimizer, etc.

3.2. Graph Neural Architecture Search

In general, graph NAS could be formulated as the following bi-level optimization problem:

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha} \in \mathcal{O}}{\arg\max} \operatorname{Acc}_{val}(\boldsymbol{W}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \tag{3}$$

$$\text{s. t. } \boldsymbol{W}^*(\boldsymbol{\alpha}) = \underset{\boldsymbol{W}}{\arg\min} \mathcal{L}_{train}(\boldsymbol{W}, \boldsymbol{\alpha}), \tag{4}$$

where Eq. (3) is the upper-level optimization problem to find the best architecture $\boldsymbol{\alpha}^*$ in the search space $\mathcal{O}$, and Eq. (4) is the lower-level problem to calculate optimal model weights $\boldsymbol{W}$ for one particular architecture $\boldsymbol{\alpha}$. Acc_{val} represents the prediction accuracy on the validation set and $\mathcal{L}_{train}$ is the classification cross-entropy loss on the training set.

Existing graph NAS methods [1, 10, 12, 48, 50] design their search space following the message-passing scheme in Eq. (1). The most commonly used correlation coefficient operations are provided in Appendix B.2. NAS methods can search all components in Eq. (1) such as the aggregation function, correlation coefficients, and activation functions as well as hyper-parameters like hidden size and learning configurations. In this paper, we mainly consider searching for architectural designs.

4. Robust Graph Neural Architecture Search

In this section, we first formulate the problem of graph robust NAS. Then, we introduce our novel and expressive search space with defensive operation candidates, namely graph structure masks. Based on the designed search space, we build a supernet containing all possible architectures and train it in a single-path one-shot way. Finally, we introduce our proposed robustness metric and describe the search process exploiting the evolutionary algorithm.

4.1. Problem formulation

Given a search space $\mathcal{O}$, we aim to find the optimal architecture $\boldsymbol{\alpha}^* \in \mathcal{O}$ with both high prediction accuracy and high adversarial robustness. We formulate the problem of robust neural architecture search for GNNs as:

$$\boldsymbol{\alpha}^* = \arg\max_{\boldsymbol{\alpha}\in\mathcal{O}} \text{ACC}_{val}(\boldsymbol{\alpha}) + \lambda\mathcal{R}(\boldsymbol{\alpha}), \tag{5}$$

where $\mathcal{R}(\cdot)$ is the robustness metric, and λ is a hyper-parameter balancing the model accuracy and robustness.

4.2. Search Space for Robust GNNs

We design a fine-grained search space following the message-passing paradigm. In total, there are six adjustable components in our GNN architecture: the graph structure mask, the nodes correlation coefficient, the neighbor aggregator, the combine function, the skip connection, and the layer aggregator. The first four components belong to the intra-layer operations, while the rest two components are inter-layer operations.

Intra-layer Operations. Inside the l-th message-passing layer, the defensive operation $\mathcal{D} \in \mathcal{O}_{\mathcal{D}}$ is firstly adopted to construct a graph structure mask and reconstruct the graph structure:

$$\mathbf{M}^{(l)} = \mathcal{D}(\mathbf{A}^{(l-1)}), \mathbf{A}^{(l)} = \mathbf{A} \odot \mathbf{M}^{(l)} \tag{6}$$

where $\mathbf{M}^{(l)} = \{m_{ij}\}$ is the graph structure mask matrix and $\mathbf{A}^{(l)}$ denotes the graph structure in the l-th layer. $\odot$ is the Hadamard product and $\mathbf{A}^{(0)} = \mathbf{A}$. Each element $m_{ij} \in [0, 1]$ is the mask score between node i and node j where $m_{ij} = 0$ indicates a complete edge pruning and $m_{ij} = 1$ means no modification to the original edge. The defensive operation aims to assign fewer weights to potential perturbed edges.

Table 1. Graph structure mask operations $O_{\mathcal{D}}$. The detailed denotation is introduced in Appendix B.1.

$\mathcal{O}_{\mathcal{D}}$	**Formula**
Identity	$\mathbf{M}^{(l)} = \mathbf{A}^{(l-1)}$
LRA	$\mathbf{A}^{(l-1)} = \mathbf{U}^{(l-1)}\mathbf{S}^{(l-1)}(\mathbf{V}^{(l-1)})^T$, $\mathbf{M}^{(l)} = \mathbf{U}_r^{(l-1)}\mathbf{S}_r^{(l-1)}(\mathbf{V}_r^{(l-1)})^T$
NFS	$m_{ij}^{(l)} = \begin{cases} 0, & if\ a_{ij}^{(l-1)} > 0\ and\ J_{ij} < \tau \\ a_{ij}^{(l-1)}, & otherwise \end{cases}$
NIE	$m_{ij}^{(l)} = \beta m_{ij}^{(l-1)} + (1-\beta)\hat{\alpha}_{ij}^{(l)}$
VPO	$\mathbf{M}^{(l)} = \sum_{v=1}^{V} \theta_v (\mathbf{A}^{(l-1)})^v$

Inspired by the success of current defensive approaches [33], we conclude the properties of operations on graph structure for robustness and design representative defensive operations in our search space accordingly. In this way, we can choose the most appropriate defensive strategies when confronting perturbed graphs. To our best knowledge, this is the first time the search space to be designed with a specific purpose to enhance the robustness of GNNs. Specifically, we include five graph structure mask operations in the search space. *Identity* keeps the same graph structure as the previous layer. *L*ow *R*ank *A*pproximation (*LRA*) reconstructs the adjacency matrix in the l-th layer from the top-r components of singular value decomposition from adjacency matrix in the previous layer. *N*ode *F*eature *S*imilarity (*NFS*) deletes edges that have small Jaccard similarities among node features. *N*eighbor *I*mportance *E*stimation (*NIE*) updates mask values with a pruning strategy based on quantifying the relevance among nodes. *V*ariable *P*ower *O*perator (*VPO*) forms a variable power graph from the original adjacency matrix weighted by the parameters of influence strengths.

Moreover, more operations like graph structure learning [52] could be integrated into the graph structure mask operations. The formula for mask operations is shown in Table 1. We limit *LRA* and *NFS* in the first layer (only for pre-processing) and exploit the other three operations for all layers. It is worth mentioning that the graph structure mask candidates could be easily extended to other methods that deal with the graph structure, like denoising methods. Mask operations will not occlude the later message-passing process. They could be seen as orthogonal operations to other operations like the correlation coefficient.

For the other three components in the intra-layer, our choices for correlation coefficients follow the literature summarized in Table 4 in Appendix. Based on the masked graph

Table 2. The six components in search space and corresponding candidate operations.

Component	Candidate Operations
$\mathcal{O}_{\mathcal{D}}$	*Identity, LRA, NFS, VPO, NIE*
$\mathcal{O}_{e}$	*Identity, GCN, GAT, GAT-Sym, Cos, Linear, Gene-Linear*
$\mathcal{O}_{aggr}$	*Sum, Mean, Max*
$\mathcal{O}_{comb}$	*Identity, GIN, SAGE*
$\mathcal{O}_{skip}$	*Identity, Zero*
$\mathcal{O}_{layer}$	*Concat, Max, LSTM*

structure $\mathbf{A}^{(l)}$ and correlation coefficients $\{e_{ij}^{(l)}\}$, a neighbor aggregator $\mathrm{Aggr} \in \mathcal{O}_{aggr}$ is used to aggregate neighborhood representations. Afterwards, a combine function $\mathrm{Comb} \in \mathcal{O}_{comb}$ merges self- and neighbor-message. Here, we explore two typical approaches to combine self-representation and neighbor-message, namely *GraphSAGE* [15] and *GIN* [40]. *GraphSAGE* conducts different feature transformations for node representation and neighborhood information. *GIN* first performs weighted sum for self- and neighbor-message, then uses multi-layer perceptron to improve GNN's expressive power. After combining messages, the activation function $\sigma(\cdot)$ is applied. Overall, the hidden representation for node i in the l-th layer is calculated as

$$\mathbf{h}_i^{(l)} = \sigma\left(\mathbf{W}^{(l)}\,\mathrm{Comb}\left(\mathbf{h}_i^{(l-1)}, \mathrm{Aggr}(m_{ij}^{(l)} e_{ij}^{(l)} \mathbf{h}_j^{(l-1)}, j \in \tilde{\mathcal{N}}(i))\right)\right). \tag{7}$$

The node representation is initialized as node features, i.e., $\mathbf{h}^{(0)} = \mathbf{X}$.

Inter-layer Operations. Following the idea of JK-Net [41] and SANE [49], we aggregate node representations in intermediate layers via the layer aggregator $\mathrm{Layer\_Aggr} \in \mathcal{O}_{layer}$. The skip operation $\mathrm{Skip} \in \mathcal{O}_{skip}$ decides the skip connection to the final layer aggregator.

$$\mathbf{z}_i = \mathrm{Layer\_Aggr}\left(\mathrm{Skip}(\mathbf{h}_i^{(1)}), ..., \mathrm{Skip}(\mathbf{h}_i^{(L)})\right), \tag{8}$$

where L is the maximum number of message-passing layers. With the JK-Net backbone, we choose the optimal model depth by setting a maximal depth. For example, if we set $L = 4$ and the skip operations are [*Idenity, Zero, Idenity, Zero*], then the selected optimal number of message-passing layers is 3. After obtaining the final node representations, we add several fully connected layers to conduct classification.

We summarize all components and their candidate operations in Table 2. Our search space is expressive and, more importantly, has improved defense capability. As shown in Table 5 in Appendix, our search space could recover some classic manually designed GNNs and state-of-the-art robust GNNs like GCN-SVD, GCN-Jaccard, GNN-Guard, and VPN.

Supenet Training. With the proposed search space, we construct a supernet and train it in a single-path one-shot way [14]. The supernet we built contains all possible architectures based on our search space, also called the one-shot model. In the supernet, each architecture behaves as a single path. Unlike graph NAS methods such as GNAS [1] or DSS [24] that use continuous relaxation for architecture parameters, we train the supernet via uniform single path sampling [14]. During the training process, we uniformly sample one path to update weights each time. Afterward, we search for optimal architectures through the evolutionary algorithm without further training steps.

4.3. Measuring Robustness

This part will introduce how to measure the architecture's robustness and the specific search process using the evolutionary algorithm with our robustness metric.

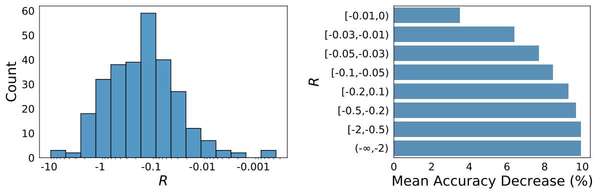
Intuitively, the performance of a robust GNN should not deteriorate too much when confronting various perturbed graph data. Let $\mathcal{T}_\Delta$ denote the attacker of the graph data with perturbation budgets Δ. We define the robustness metric as

$$\mathcal{R}(\mathbf{A}, f) = -\mathbb{E}_{\mathbf{A}'}\left[\frac{1}{N}\sum_{i=1}^{N} D_{KL}\left(f(\mathbf{A})_i || f(\mathbf{A}')_i\right)\right], \mathbf{A}' = \mathcal{T}_\Delta(\mathbf{A}), \tag{9}$$

where f indicates the GNN, $f(\mathbf{A})_i$ is the probability prediction vector for node i and $D_{KL}(p||q) = \sum_i p_i \log \frac{p_i}{q_i}$ denotes the Kullback-Leibler (KL) divergence between two distribution p and q. Here, we use KL distance to measure the prediction difference given clean and perturbed data. The choice of our attack proxy $\mathcal{T}$ varies from simple random attack [5] or DICE [54] to advanced attack methods like Mettack [54] or Nettack [53]. Within the same perturbation budgets Δ, our attack proxy generates T perturbed graph structure $\{\mathbf{A}'_t\}_{t=1}^{T}$. Eq. (9) could be approximately computed as

$$\mathcal{R}(\mathbf{A}, f) \approx -\frac{1}{TN}\sum_{t=1}^{T}\sum_{i=1}^{N}\left(D_{KL}(f(\mathbf{A})_i || f(\mathbf{A}'_t)_i)\right). \tag{10}$$

A larger $\mathcal{R}(\mathbf{A}, f)$ indicates a smaller distance between the prediction of clean data and the perturbed data, and consequently, more robust GNN architectures.



(a)

(b)

Figure 2. Evaluation of the robustness metric $\mathcal{R}$. (a) A histogram for the robustness metric in log scale. (b) The relationship between the robustness metric and accuracy decreases (%) under attacks.

In order to illustrate the effectiveness of the proposed robustness metric, we randomly choose 300 architectures and calculate their robustness metric value and evaluate their performance decrease after 5% structural perturbations in the Cora dataset (the experimental details are described in Sec. 5.2). Fig. 2a displays the distribution of $\mathcal{R}$ in log scale while Fig. 2b shows the mean accuracy decrease for different $\mathcal{R}$ intervals. The accuracy decrease could be regarded as a ground-truth measurement of the actual robustness. From Fig. 2b, we could see that the mean accuracy decrease has a negative correlation with respect to the robustness metric. When the robustness metric is relatively small, the accuracy may be reduced by as large as 10%. However, when the robustness metric is large, the accuracy decrease shrinks to about 3.5%. This phenomenon indicates that our robustness metric could successfully filter robust architectures.

Evolutionary Search Algorithm. In our work, we adopt an evolutionary algorithm to search for optimal robust graph architectures with the proposed robustness metric. Instead of training each candidate architecture from scratch, we leverage the evolutionary algorithm only for inference and search. The weights for all architectures are fixed as those learned in the supernet training phase. In one search epoch, we select top-k robust candidates via the fitness function $\mathrm{ACC}_{val}(\boldsymbol{\alpha}) + \lambda\mathcal{R}(\boldsymbol{\alpha})$. Crossover and mutation are followed to generate children architectures from population candidates. We show the search algorithm in Appendix B.4.

5. Experiments

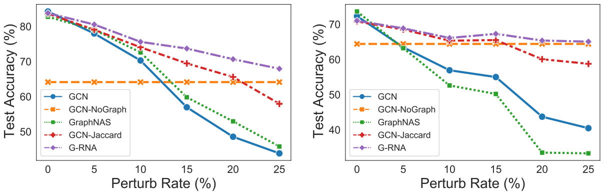
In this section, we conduct experiments to verify our proposed method by evaluating the selected architecture on perturbed graphs. Also, we visualize the performance for diverse operations to better understand GNN architecture's robustness. Additional experimental results including defensive performance under targeted attack, evaluation on heterophily graphs, and sensitivity analysis for the hyper-parameter λ in our proposed robustness metric could be found in Appendix C. The details of the experimental setting are deferred to the Appendix D.

5.1. Semi-supervised Node Classification Task

In the semi-supervised node classification task, we perform non-targeted poisoning structural attacks adopting Mettack [54] and evaluate GNN robustness based on perturbed graphs. We vary the proportion of changed edges from 0% to 25% and calculate the retrained accuracy on perturbed graphs. For each setting, we experiment 10 times and report the average results and the standard deviation. The final defense results are summarized in Table 3. More information on searched architectures by G-RNA under attacks is provided in Appendix C.3.

Overall, our G-RNA successfully outperforms all baselines under adversarial attacks. We could see an apparent increase in perturbation performance when confronting heavily poisoning attacks (*e.g.* when the proportion of changed edges is 15% or more). For instance, the defense performance of G-RNA under 25% structural perturbations exceeds that of vanilla GCN by 23.4%, 20.9%, and 12.1%, respectively. For NAS-based methods, optimal architectures searched on clean data are evaluated on both clean and perturbed graphs to test their robustness and conduct a fair comparison. We could see that there does exist a trade-off between architecture accuracy and robustness. Though GraphNAS shows impressive results on clean graphs, its vulnerability under adversarial attacks is also obvious, *e.g.* GraphNAS's performance under adversarial attacks is even worse than that of classic GNNs sometimes. Thus, it's necessary to make Graph NAS more robust. In CiteSeer, the performance for G-RNA under 20% and 25% structural perturbations are almost twice that of GraphNAS, which shows the strong defense ability of our method.

Ablation study on robustness metric. We add an ablation study by removing the robustness metric (denoted as G-RNA w/o rob). Our full G-RNA shows better performance compared to G-RNA w/o rob on all datasets under attacks, demonstrating the effectiveness of our proposed robustness metric.



(a) Cora.

(b) CiteSeer.

Figure 3. Comparison with GCN-NoGraph. G-RNA is the only method that consistently outperforms GCN-NoGraph.

Comparison with not using structures. When the graph structure is heavily poisoned, aggregating neighbor messages by the message-passing paradigm may not be reliable. As a result, we compare our method with GCNs not using edges, named GCN-NoGraph [20], i.e., a two-layer MLP on node features. Notice that the performance exceeding GCN-NoGraph indicates an effective graph structure, while that below GCN-NoGraph implies an informationless or confusing graph structure for GNNs. Fig. 3 shows the comparison with GCN-NoGraph. For all comparing methods, only the performance of G-RNA always outperforms GCN-NoGraph. GCN-Jaccard performs well for small perturbations but fails to defend against large perturbations. Also, we could see an obvious drop for GCN and GraphNAS when the perturb rate goes beyond 15% for both datasets.

5.2. Understand Operation Robustness

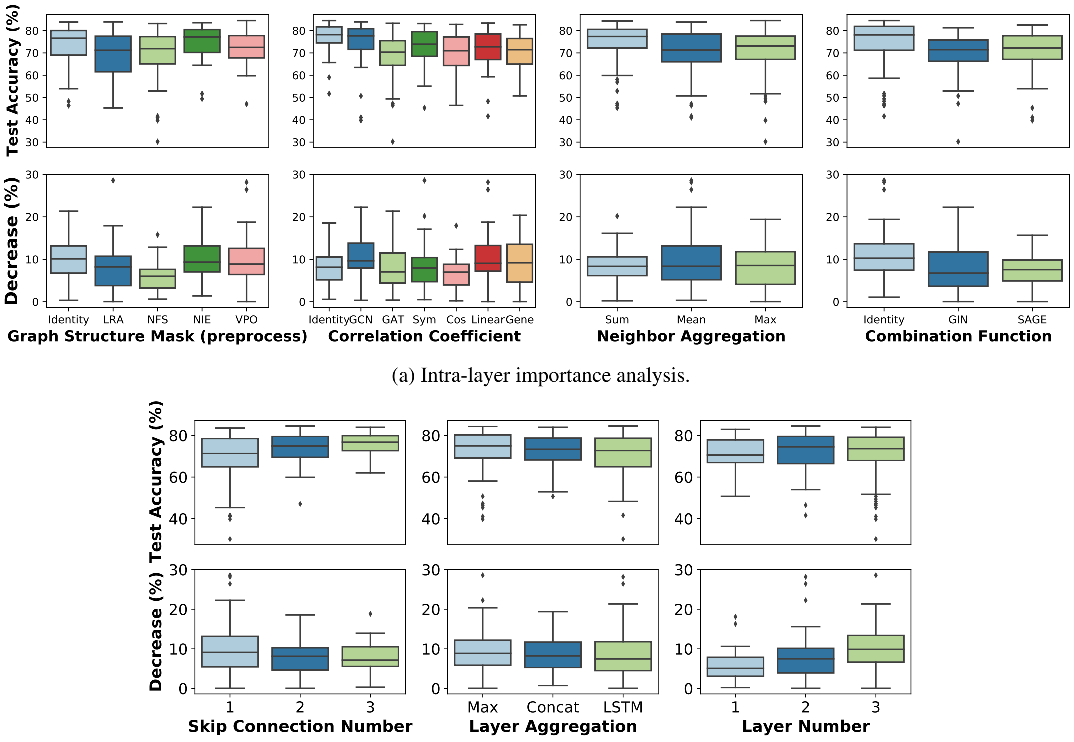
For understanding the robustness of various GNN components, we randomly sample 300 architectures and evaluate

Table 3. The results of node classification accuracy (mean±std, in percentages) under non-targeted attacks (Mettack). **Bold** numbers indicate the best performance. "-" indicates the result is unavailable due to the high time complexity of the model.

Dataset	Model		Proportion of changed edges (%)					
			0	5	10	15	20	25
Cora	Vanilla GNN	GCN	84.28±0.25	78.00±1.20	70.31±1.24	56.97±1.20	48.56±2.66	43.83±1.47
		GCN-JK	84.38±0.38	74.63±0.60	67.32±0.80	53.26±1.17	45.29±2.49	38.63±1.03
		GAT	84.30±0.65	78.56±1.33	70.18±1.43	58.39±2.27	49.35±1.55	42.40±1.02
		GAT-JK	84.16±0.36	74.30±1.35	68.07±1.12	54.54±2.55	51.20±1.51	43.29±1.47
	Robust GNN	RGCN	84.60±0.37	75.92±1.01	72.94±0.40	59.97±0.50	52.50±0.38	46.47±0.91
		GCN-Jaccard	83.64±0.76	77.07±0.61	74.07±0.59	68.92±0.80	63.57±0.87	56.14±1.45
		Pro-GNN	84.64±0.59	79.59±0.83	73.73±0.76	62.10±1.65	54.89±2.03	48.98±1.89
		DropEdge	83.35±1.23	77.84±0.30	70.96±0.44	57.13±0.50	49.70±0.72	43.51±1.13
		PTDNet	83.70±0.43	78.49±0.43	70.94±0.34	54.39±0.81	45.80±0.63	41.32±0.67
	Graph NAS	GraphNAS	82.77±0.40	72.97±2.34	57.12±5.31	44.50±1.48	37.21±3.79	31.96±1.68
		GASSO	84.11±0.34	77.69±1.10	68.50±0.64	56.61±0.90	51.87±0.79	46.05±2.01
		G-RNA w/o rob	**84.29±0.40**	77.39±1.38	67.61±1.73	53.56±3.00	48.57±1.85	41.20±1.53
		G-RNA	83.81±0.39	**80.45±0.74**	**75.16±0.89**	**73.52±0.86**	**70.6±1.43**	**67.23±1.66**
CiteSeer	Vanilla GNN	GCN	72.35±0.49	63.48±0.45	56.94±1.29	55.01±0.91	43.73±1.15	40.47±0.77
		GCN-JK	72.14±0.36	62.16±0.51	54.25±1.30	50.61±0.93	40.06±1.11	35.07±1.51
		GAT	71.75±0.71	61.47±1.33	53.92±1.29	49.68±2.74	41.31±2.95	37.25±2.26
		GAT-JK	71.52±0.90	63.90±0.38	57.16±0.53	52.80±1.08	44.26±0.87	39.63±0.69
	Robust GNN	RGCN	72.43±0.41	63.30±0.33	55.90±0.47	53.83±0.31	43.65±0.72	39.99±0.46
		GCN-Jaccard	71.03±0.45	64.56±0.75	57.57±0.74	55.31±0.82	50.17±0.66	45.78±0.43
		Pro-GNN	71.74±0.76	66.29±0.64	64.60±0.86	63.96±1.48	62.46±1.12	55.73±2.04
		DropEdge	71.84±0.21	63.33±0.70	55.41±0.77	50.81±0.93	40.61±0.92	37.46±0.61
		PTDNet	**72.87±0.46**	64.08±0.78	57.03±0.37	54.04±0.75	41.81±0.63	38.93±0.66
	Graph NAS	GraphNAS	72.79±0.22	61.01±0.36	53.55±1.29	55.98±12.01	39.06±6.79	36.90±4.76
		GASSO	71.08±0.29	61.31±0.53	52.17±0.70	50.43±0.87	43.72±1.10	36.84±0.55
		G-RNA w/o rob	72.57±0.29	65.48±1.29	56.59±1.76	55.81±1.59	48.53±2.20	43.76±2.23
		G-RNA	71.32±0.82	**68.71±1.20**	**65.84±1.20**	**65.29±1.35**	**62.58±0.99**	**61.33±1.35**
PubMed	Vanilla GNN	GCN	86.35±0.15	82.70±0.13	80.56±0.16	77.85±0.17	75.85±0.18	73.68±0.22
		GCN-JK	87.07±0.12	82.76±0.15	81.56±0.18	80.22±0.38	79.14±0.44	77.31±0.24
		GAT	85.28±0.20	81.02±0.31	79.58±0.16	76.39±0.43	74.41±0.20	72.22±0.24
		GAT-JK	85.72±0.14	82.37±0.10	80.60±0.23	78.50±0.15	76.39±0.14	74.02±0.25
	Robust GNN	RGCN	86.64±0.08	82.90±0.18	80.73±0.19	77.86±0.17	75.89±0.15	73.74±0.22
		GCN-Jaccard	87.11±0.04	83.95±0.06	82.30±0.08	80.16±0.07	78.83±0.13	76.86±0.17
		Pro-GNN	-	-	-	-	-	-
		PTDNet	83.87±0.24	74.32±0.44	68.80±0.34	67.32±0.18	66.50±0.12	65.21±0.34
		DropEdge	83.93±0.10	83.24±0.12	82.33±0.15	81.06±0.18	79.21±0.14	76.88±0.28
	Graph NAS	GraphNAS	87.26±0.04	83.56±0.08	80.00±3.98	77.86±2.59	72.97±3.88	68.05±2.26
		GASSO	86.27±0.12	84.15±0.15	83.18±0.21	82.56±0.25	81.73±0.36	83.25±1.26
		G-RNA w/o rob	87.18±0.07	82.59±0.14	80.29±0.17	78.11±0.24	75.98±0.33	73.60±0.25
		G-RNA	**87.48±0.12**	**87.01±0.11**	**86.5±0.14**	**86.04±0.21**	**85.94±0.18**	**85.82±0.12**

their performance on clean and attacked graphs. To make a fair comparison, we keep the same training configurations for all selected architectures. Specifically, we train each architecture for 200 epochs with a learning rate of 0.005 and a weight decay of 5e-4. We use the decrease in classification accuracy as the measurement and report the results on the Cora dataset under Mettack (5% perturbation rate), while other datasets show similar trends. The less the performance of one architecture decreases, the more robust that architecture is. Fig. 4a and Fig. 4b show boxplots for the robustness under various architectural designs. For each subplot, the top figure displays the relationship between the operation choice and model test accuracy on the clean graph, while the bottom one shows the operation robustness via accuracy decreases. We make the following observations.

Intra Message-Passing Layer Design. Fig. 4a shows the architecture's robustness for different intra-layer architecture designs. For simplicity, we only consider intra-layer operations for the first layer. We could see that architectural design plays a significant role in both architectural accuracy



(a) Intra-layer importance analysis.

(b) Inter-layer importance analysis.

Figure 4. The importance of intra-layer and inter-layer designs to GNN robustness. The first row of each picture is the clean test accuracy (%) for various operations(higher is better), while the second row shows the accuracy decrease (%) under perturbations (lower is better).

and robustness. The use of pre-processing graph structure mask operations increases GNN models' robustness but sacrifices some accuracy. For the Cora dataset, *NFS* is the most effective operation for pre-processing perturbed graph data. *GCN* is the most fragile correlation coefficient under Mettack. A plausible reason is that Mettack adopts a two-layer GCN model as the surrogate model to generate adversarial samples. For neighbor aggregation operations, *Sum* shows high accuracy and relatively robust performance. What's more, *Max* is also able to generate some robust architectures. Besides, combination functions are essential to enhance the robustness of GNNs as we could see a smaller accuracy decrease when using *GIN* or *SAGE* operations. Consequently, it is necessary to distinguish self- and neighbor-messages in the message-passing.

Inter Message-Passing Layer Design. Our inter-layer designs include skip connections and layer aggregation operations. We also study how the number of message-passing layers affects the robustness of GNNs. The results are summarized in Fig. 4b. Skip connections help elevate both model accuracy and robustness. For the layer aggregation, there are no obvious differences, and *LSTM* behaves slightly more robustly than the other two operations. More interestingly, increasing the model depth makes GNNs more fragile under adversarial attacks. A 2-layer GNN shows reasonably good performance for the Cora dataset.

6. Conclusion and Limitations

In this paper, we propose the first adversarially robust NAS framework for GNNs. We incorporate graph structure mask operations into the search space to enhance the defensive ability of GNNs. We also define a robustness metric that could effectively measure the architecture's robustness. Experimental results demonstrate the effectiveness of our proposed approaches under adversarial attacks. We empirically analyze the architectural robustness for different operations, which provides insights into the robustness mechanism behind GNN architectures.

Meanwhile, the lack of efficiency is a shared issue for many NAS methods. Since this weakness is not our main focus, we would like to leave this limitation as a future work. Societal impacts are discussed in the Appendix.

Acknowledgements.

This work was supported in part by the National Key Research and Development Program of China No. 2020AAA0106300, National Natural Science Foundation of China (No. 62250008, 62222209, 62102222, 61936011, 62206149), Beijing National Research Center for Information Science and Technology (BNRist) under Grant No. BNR2023RC01003, Beijing Key Lab of Networked Multimedia, China National Postdoctoral Program for Innovative Talents No. BX20220185, and China Postdoctoral Science Foundation No. 2022M711813.

References

[1] Shaofei Cai, Liang Li, Jincan Deng, Beichen Zhang, Zheng-Jun Zha, Li Su, and Qingming Huang. Rethinking graph neural architecture search from message-passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6657–6666, 6 2021. 1, 3, 4, 5

[2] Heng Chang, Yu Rong, Tingyang Xu, Yatao Bian, Shiji Zhou, Xin Wang, Junzhou Huang, and Wenwu Zhu. Not all low-pass filters are robust in graph convolutional networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021. 2

[3] Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Honglei Zhang, Peng Cui, Xin Wang, Wenwu Zhu, and Junzhou Huang. Adversarial attack framework on graph embedding models with limited knowledge. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2022. 2

[4] Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Honglei Zhang, Peng Cui, Wenwu Zhu, and Junzhou Huang. A restricted black-box adversarial framework towards attacking graph embedding models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3389–3396, 2020. 2

[5] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR, 2018. 5

[6] Zulong Diao, Xin Wang, Dafang Zhang, Yingru Liu, Kun Xie, and Shaoyao He. Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 890–897, 2019. 1

[7] Yuhui Ding, Quanming Yao, Huan Zhao, and Tong Zhang. Diffmg: Differentiable meta graph search for heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, page 279–288, 2021. 3

[8] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 169–177, 2020. 1, 2

[9] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 15

[10] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graph neural architecture search. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 1403–1409, 2020. 1, 3, 4, 15

[11] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *International Conference on Machine Learning*, pages 1263–1272, 2017. 3

[12] Chaoyu Guan, Ziwei Zhang, Haoyang Li, Heng Chang, Zeyang Zhang, Yijian Qin, Jiyan Jiang, Xin Wang, and Wenwu Zhu. Autogl: A library for automated graph learning. In *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*, 2021. 4

[13] Minghao Guo, Yuzhe Yang, Rui Xu, Ziwei Liu, and Dahua Lin. When nas meets robustness: In search of robust architectures against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 631–640, 2020. 3

[14] Zichao Guo et al. Single path one-shot neural architecture search with uniform sampling. In *ECCV*, 2020. 2, 3, 5

[15] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *neural information processing systems*, pages 1024–1034, 2017. 5, 11

[16] Ramtin Hosseini, Xingyi Yang, and Pengtao Xie. Dsrna: Differentiable search of robust neural architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6196–6205, 2021. 3

[17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020. 14

[18] Hongwei Jin, Zhan Shi, Venkata Jaya Shankar Ashish Peruri, and Xinhua Zhang. Certified robustness of graph convolution networks for graph classification under topological attacks. *Advances in Neural Information Processing Systems*, 33, 2020. 3

[19] Ming Jin, Heng Chang, Wenwu Zhu, and Somayeh Sojoudi. Power up! robust graph convolutional network via graph powering. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pages 8004–8012, 2021. 2, 11

[20] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 66–74, 2020. 2, 6, 14

[21] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*, 2017. 11, 14

[22] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9267–9276, 2019. 3

[23] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020. 15

[24] Yanxi Li, Zean Wen, Yunhe Wang, and Chang Xu. One-shot graph neural architecture search with dynamic search space. *AAAI*, 35(10):8510–8517, May 2021. 1, 3, 5

[25] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015. 1

[26] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018. 3

[27] Dongsheng Luo et al. Learning to drop: Robust graph neural network via topological denoising. In *WSDM*, 2021. 15

[28] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52, 2015. 14

[29] Zheyi Pan, Songyu Ke, Xiaodu Yang, Yuxuan Liang, Yong Yu, Junbo Zhang, and Yu Zheng. Autostg: Neural architecture search for predictions of spatio-temporal graph. In *Proceedings of the Web Conference 2021*, page 1846–1855, 2021. 3

[30] Yijian Qin et al. Graph differentiable architecture search with structure learning. *NeurIPS*, 2021. 3, 15

[31] Yu Rong et al. Dropedge: Towards deep graph convolutional networks on node classification. *ICLR*, 2020. 15

[32] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *Ai Magazine*, 29(3):93–106, 2008. 14

[33] Lichao Sun, Ji Wang, Philip S. Yu, and Bo Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018. 1, 2, 4

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR 2018 : International Conference on Learning Representations 2018*, 2018. 11, 14

[35] Binghui Wang, Jinyuan Jia, Xiaoyu Cao, and Neil Zhenqiang Gong. Certified robustness of graph neural networks against adversarial structural perturbation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1645–1653, 2021. 3

[36] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. A semi-supervised graph attentive network for financial fraud detection. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 598–607. IEEE, 2019. 1

[37] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, pages 2022–2032, 2019. 1

[38] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4816–4823. AAAI Press, 2019. 1, 2, 14

[39] Beini Xie, Heng Chang, Xin Wang, Tian Bian, Shiji Zhou, Daixin Wang, Zhiqiang Zhang, and Wenwu Zhu. Revisiting adversarial attacks on graph neural networks for graph classification. *arXiv preprint arXiv:2208.06651*, 2022. 2

[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks. In *ICLR 2019 : 7th International Conference on Learning Representations*, 2019. 5, 11

[41] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. *international conference on machine learning*, pages 5449–5458, 2018. 3, 5, 11, 14

[42] Jiaxuan You, Rex Ying, and Jure Leskovec. Design space for graph neural networks. In *NeurIPS*, 2020. 3

[43] Junchi Yu, Jie Cao, and Ran He. Improving subgraph recognition with variational graph information bottleneck. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19396–19405, 2022. 2

[44] Junchi Yu, Tingyang Xu, Yu Rong, Yatao Bian, Junzhou Huang, and Ran He. Graph information bottleneck for subgraph recognition. In *International Conference on Learning Representations*, 2021. 2

[45] Junchi Yu, Tingyang Xu, Yu Rong, Junzhou Huang, and Ran He. Structure-aware conditional variational auto-encoder for constrained molecule optimization. *Pattern Recognition*, 126:108581, 2022. 1

[46] Xiang Zhang and Marinka Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 2, 11

[47] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020. 2

[48] Ziwei Zhang, Xin Wang, and Wenwu Zhu. Automated machine learning on graphs: A survey. In *IJCAI*, 2021. 1, 3, 4

[49] Huan Zhao, Quanming Yao, and Weiwei Tu. Search to aggregate neighborhood for graph neural network. In *ICDE*, 2021. 1, 3, 5

[50] Kaixiong Zhou, Qingquan Song, Xiao Huang, and Xia Hu. Auto-GNN: Neural Architecture Search of Graph Neural Networks. *arXiv e-prints*, page arXiv:1909.03184, 2019. 1, 4

[51] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 1399–1407, 2019. 1, 2, 14

[52] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Yuanqi Du, Jieyu Zhang, Qiang Liu, Carl Yang, and Shu Wu. A Survey on Graph Structure Learning: Progress and Opportunities. *arXiv e-prints*, page arXiv:2103.03036, 2021. 4

[53] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018. 1, 2, 5, 11

[54] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations (ICLR)*, 2019. 5, 6

[55] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856, 2018. 1