A Comprehensive Survey on Trustworthy Graph Neural Networks: Privacy, Robustness, Fairness, and Explainability

Enyan Dai[1]   Tianxiang Zhao[1]   Huaisheng Zhu[1]   Junjie Xu[1]

Zhimeng Guo[1]   Hui Liu[2]   Jiliang Tang[2]   Suhang Wang[1]

[1]The Pennsylvania State University, State College 16801, USA

[2]Michigan State University, East Lansing 48824, USA

Abstract: Graph neural networks (GNNs) have made rapid developments in the recent years. Due to their great ability in modeling graph-structured data, GNNs are vastly used in various applications, including high-stakes scenarios such as financial analysis, traffic predictions, and drug discovery. Despite their great potential in benefiting humans in the real world, recent study shows that GNNs can leak private information, are vulnerable to adversarial attacks, can inherit and magnify societal bias from training data and lack interpretability, which have risk of causing unintentional harm to the users and society. For example, existing works demonstrate that attackers can fool the GNNs to give the outcome they desire with unnoticeable perturbation on training graph. GNNs trained on social networks may embed the discrimination in their decision process, strengthening the undesirable societal bias. Consequently, trustworthy GNNs in various aspects are emerging to prevent the harm from GNN models and increase the users′ trust in GNNs. In this paper, we give a comprehensive survey of GNNs in the computational aspects of privacy, robustness, fairness, and explainability. For each aspect, we give the taxonomy of the related methods and formulate the general frameworks for the multiple categories of trustworthy GNNs. We also discuss the future research directions of each aspect and connections between these aspects to help achieve trustworthiness.

Keywords: Graph neural networks (GNNs), trustworthy, privacy, robustness, fairness, explainability.

1 Introduction

Graph-structured data such as bioinformatics network[1], trading network[2], and social network[3] are pervasive in the real-world. Inspired by the great success of deep learning on independent and identically distributed (i.i.d) data such as images, graph neural networks (GNNs)[4–7] are investigated to generalize deep neural networks to model graph-structured data. GNNs have shown great performance for various applications across various domains including finance[8, 9], healthcare[10] and social analysis[11, 12]. The success of GNNs relies on the message-passing mechanism, where node representations are updated by aggregating the information from neighbors. With this mechanism, node representations can capture node features, information of neighbors and local graph structure, which facilitate various graph mining tasks, such as node classification[4], link prediction[13] and graph classification[14].

Despite their achievements in modeling graphs, the concerns in the trustworthiness of GNNs are rising. Firstly, GNN models are vulnerable to the attacks that steal the private data information or affect the behaviors of the model. For example, hackers can utilize the embeddings of nodes to infer their attribute information and friendship information in social network[15, 16]. They also can easily fool the GNNs to give target prediction to a node by injecting malicious nodes to the network[17]. Secondly, GNN models themselves have problems in fairness and interpretability. More specifically, GNN models can magnify the bias in the training data, resulting discrimination towards the people with certain genders, skin colors, and other protected sensitive attributes[18, 19]. Finally, due to the high nonlinearity of the model, predictions from the GNNs are difficult to understand. The lack of interpretability also makes the GNNs untrustworthy, which largely limit the applications of GNNs. Those

weaknesses significantly hinder the adoption of GNNs in real-world applications, especially those high-stake scenarios such as finance and healthcare. Therefore, how to build trustworthy GNN models has become a focal topic.

Recently, a guideline for trustworthy AI systems have been proposed by the European Union[20]. As shown in Fig. 1, the guideline indicates that trustworthy AI should obey the following four ethical principles: respect for human autonomy, prevention of harm, fairness, and explainability. The principle of respect for human autonomy requires AI systems to follow human-centric design principles and leave meaningful opportunity for human choice. This generally fails in the domain of human-computer interaction. Therefore, we do not focus on this direction of trustworthy GNNs in this survey. According to the principle of prevention of harm, AI systems should be technically robust and be ensured not open to malicious use, which corresponds to the robustness and privacy aspects of our survey. The principle of fairness requires that AI systems should ensure the individuals and groups free from unfair bias, discrimination and stigmatisation. As for the explainability, it requires the decision process of AI to be transparent and explainable. It is worth to mention that the four aspects are not isolated with each other. For instance, the attacker may poison the training data to degrade the fairness of the model[21, 22] or mislead the GNN explainer model[23]. And the explanations from explainable GNN methods can also be helpful for other aspects. Specifically, based on the explanations from the explainable GNN model, the human can debug the model to avoid the adversarial attacks. In addition, with the analysis of the explanations, we can evaluate whether the deployed model is giving biased predictions. Therefore, it is important to explore the connections of these aspects to finally achieve trustworthy GNNs that simultaneously address the concerns in robustness, privacy, fairness, and explainability. In this survey, we also discuss about the interactions of the trustworthiness aspects in the future directions.

Due to the demand for trustworthy GNNs, a large number of literatures in different aspects of trustworthy GNNs are emerging in recent years. For example, robust GNN against the perturbations from attackers have been developed[30–32]. To prevent the private information, privacy-preserving GNN models[33, 34] are also proposed in various real-world applications such as financial analysis. Fair GNNs[18] and explainable GNNs[35, 36] also become hot topics to address the concerns in trustworthiness. There are several surveys of GNNs in robustness[26, 37–39], explainability[28], and fairness[27]. However, none of them thoroughly discuss about the trustworthiness of GNNs, which should also cover the dimensions of privacy and fairness. For the aspects of robustness and explainability, they also do not include the emerging directions and techniques such as scalable attacks, backdoor attacks, and self-explainable GNNs, which are discussed in this survey. A recent survey[40] gives a review about the trustworthy AI systems. But it mainly focus on the techniques of trustworthy AI systems on i.i.d data. Considering the complexity of the graph topology and the deployment of message-passing mechanism in GNNs, the trustworthy AI designed for i.i.d data generally can not be adopted to process graph-structured data. There is a concurrent survey in trustworthy graph neural networks[29]. Compared with [29], we cover more recent advanced topics of trustworthiness such as machine unlearning, model ownership verification, scalable adversarial attacks, fair contrastive learning, explanation-enhanced fairness, and self-explainable GNNs. A more comprehensive comparison between the existing surveys are presented in Table 1. To summarize, our major contributions are:

● In Section 3, we give a comprehensive survey of the works in privacy attacks and defense on GNNs followed by future directions. Graph datasets in privacy area are also listed.

● Various categories of adversarial attack and defense methods on GNN models are discussed in Section 4. Some recent advances about the robustness of GNN such as scalable attacks, graph backdoor attacks, and self-supervised learning defense are further introduced.

● The fairness of trustworthy GNNs is thoroughly discussed in the Section 5 of this survey, which includes the the biases and fairness definitions on graph-structured data, various fair GNN models and the datasets they applied.

● A comprehensive survey of GNN explainability is presented in Section 6, in which we go through the motivations, challenges, and experiment settings adopted by existing works. A taxonomic summary of methodologies is also introduced.
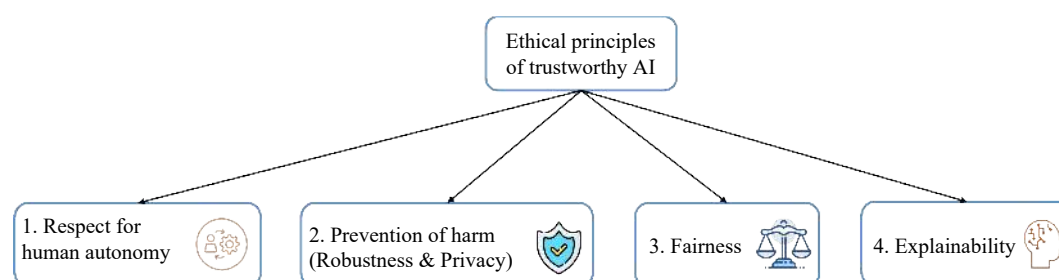

Fig. 1   The ethical principles of trustworthy AI

Table 1   Comparisons between our survey and other related surveys in trustworthy GNNs

		Zhang et al.[24]	Said et al.[25]	Jin et al.[26]	Dong et al.[27]	Yuan et al.[28]	Zhang et al.[29]	Our survey
Perspective	Privacy	●	●	○	○	○	●	●
	Fairness	○	○	●	○	○	●	●
	Robustness	○	○	○	●	○	●	●
	Explainability	○	○	○	○	●	●	●
Recent advances	Unlearning	○	●	○	○	○	○	●
	IP protection	○	●	○	○	○	○	●
	Graph backdoor	○	○	○	○	○	●	●
	Fair pretraining	○	○	○	●	○	○	●
	Explanation-enhanced fairness	○	○	○	●	○	○	●
	Self-explanation	○	○	○	○	○	●	●

2   Preliminaries

To facilitate the discussion of trustworthy GNNs, we firstly introduce notations, the basic design of GNNs, and graph analysis tasks in this section.

2.1   Notations

We use $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to denote a graph, where $\mathcal{V} = \{v_1, \cdots, v_N\}$ is the set of N nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. The graph can be either attributed or plain graph. For attributed graph, node attributes $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$ are provided, where $\boldsymbol{x}_i \in \mathbf{R}^d$ corresponds to the d-dimensional attributes of node v_i. $\boldsymbol{A} \in \mathbf{R}^{N \times N}$ is the adjacency matrix of the graph $\mathcal{G}$, where $\boldsymbol{A}_{ij} = 1$ if nodes v_i and v_j are connected; otherwise, $\boldsymbol{A}_{ij} = 0$.

2.2   Inner working of GNNs

Apart from the node features, the graph topology also offers important information for representation learning. Generally, GNNs adopt the message-passing mechanism to learn node representations that capture both node features and graph topology information. Specifically, in each layer, GNNs will update the representations of a node by aggregating the information from their neighborhood nodes. As a result, a k-layer GNN would capture the information of the local graph containing k-hop neighbors of the central nodes. The general form of updating representations in the k-th layer of GNNs can be formulated as

$$
\begin{aligned}
\boldsymbol{a}_v^{(k-1)} &= \text{AGGREGATE}^{(k-1)}\{\boldsymbol{h}_u^{(k-1)} : u \in \mathcal{N}(v)\} \\
\boldsymbol{h}_v^{(k)} &= \text{COMBINE}^{(k)}(\boldsymbol{h}_v^{(k-1)}, \boldsymbol{a}_v^{(k-1)})
\end{aligned} \tag{1}
$$

where $\boldsymbol{h}_v^{(k)}$ stands for the representation of node $v \in \mathcal{V}$ after the k-th GNN layer and $\mathcal{N}(v)$ denotes the set of neighborhoods of node v. For node classification, a linear classifier can be applied to the representation $\boldsymbol{h}_v$ to predict the label of node v. For graph classification, a READOUT function will summarize the node embeddings to a graph embedding $\boldsymbol{h}^G$ for future predictions:

$$
\boldsymbol{h}^G = \text{READOUT}(\{\boldsymbol{h}_v^K : v \in \mathcal{G}\}) \tag{2}
$$

where READOUT can be various graph pooling functions such as max pooling and average pooling. Similar to node classification, graph classification can be conducted by applying a linear classifier on the graph embedding $\boldsymbol{h}^G$.

Extensive graph neural networks that follow (1) have been proposed. Here, we only introduce the design of GCN[4], which is one of the most popular GNN architectures. For the design of other GNNs, please refer to the survey of GNN models[41]. More specifically, each layer of GCN can be written as

$$
\boldsymbol{H}^{(k)} = \sigma(\tilde{\boldsymbol{A}} \boldsymbol{H}^{(k-1)} \boldsymbol{W}^{(k)}) \tag{3}
$$

where $\boldsymbol{H}^{(k)}$ denote the representations of all the nodes after the k-th layer; $\boldsymbol{W}^{(k)}$ stands for the parameters of the k-th layer. $\tilde{\boldsymbol{A}}$ is the normalized adjacency matrix. Generally, the symmetric normalized form is used, which can be written as $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A} + \boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}$, and $\boldsymbol{D}$ is a diagonal matrix with $D_{ii} = \sum_i A_{ij}$. $\boldsymbol{I}$ is the identity matrix. σ is the activation function such as ReLU.

2.3   Graph analysis tasks

The learned node representations by GNNs can facilitate various tasks, such as node classification, link prediction, community detection and graph classification. Next, we briefly introduce them.

Node classification. Many real-world problems can be treated as the node classification problem such as user attribute prediction in social media[18, 42], fraud detection in transaction networks[2, 43], and protein function prediction on protein-protein interaction networks[44]. In node-level classification, the GNN model aims to infer the labels of the test nodes $\mathcal{V}_T$ given the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Generally, the node classification task is semi-supervised

where only partial nodes $\mathcal{V}_L \in \mathcal{V}$ are provided with labels $\mathcal{Y}$. Based on whether the test samples $\mathcal{V}_T$ are seen during the training phase, node classification can be split into transductive setting and inductive setting. In transductive setting, the test nodes $\mathcal{V}_T$ are available during the training phase. The node features of the test nodes can be utilized for better prediction performance. In contrast, in inductive setting the test nodes are totally new for the trained GNN model.

Community detection. Communities are subgraphs of a network, which are more densely connected to each other than the rest nodes in the network. Formally, the set of communities in the network can be represented by $\{\mathcal{C}_1, \cdots, \mathcal{C}_K\}$, where $\mathcal{C}_i \subset \mathcal{G}$ is a partition of the whole graph $\mathcal{G}$. These communities could be either disjointed or overlapping. The goal of the community detection is to identity which communities each node $v \in \mathcal{G}$ belongs to. Community detection is often unsupervised[45–47]. Recently, supervised community detection based on GNNs are also investigated[48]. Community detection can be useful for various domains such as social network analysis[49] and functional region identification in brain[50].

Link prediction. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the link prediction model will predict the existence of link between nodes $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$, where $(v_i, v_j) \notin \mathcal{E}$. A very common form of link prediction is to give prediction based on the representations of two nodes from a GNN model. Let $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ denote the representations of node v_i and v_j, it can be formulated as $g(v_i, v_j) = MLP(\boldsymbol{h}_i, \boldsymbol{h}_j)$. Link prediction have various applications such as friend recommendation on social media[51] and knowledge graph completion[52].

Graph classification. For graph classification, each graph instance belongs to a certain class. The training set of the graph classification can be denoted as $\mathcal{D}_T = \{(\mathcal{G}_i, y_i)\}_{i=1}^{|\mathcal{D}_T|}$, where y_i denotes the label of graph $\mathcal{G}_i$ and $|\mathcal{D}_T|$ represents the number of graphs in the training set. The goal of the graph classification is to learn a function $f_\theta : \mathcal{G} \to y$ to classify the unlabeled test graphs $\mathcal{D}_U$. As it is mentioned in Section 2.2, a READOUT function is added to the GNN model to obtain graph embedding for classification. Similarly, there are many applications of graph classification such as property prediction of drugs where each drug is represented as a graph.

3 Privacy of graph neural networks

Similar to deep learning algorithms on images and texts, the remarkable achievements of GNNs also rely on the big data. Extensive sensitive data are collected from users to obtain powerful GNN models for various services in critical domains such as healthcare[60], banking systems[2], and bioinformatics[10]. For example, GNNs have been applied on brain networks for FMRI analysis[10]. In addition, the GNN model owner may provide the query API service to share the knowledge learned by GNNs. It is also very common that the pretrained GNN models are released to the third parties for knowledge distillation or various downstream tasks[61]. However, the collection and utilization of private data for GNN model training, the API service and model release are threatening the safety of the private and sensitive information. First, GNNs are generally trained in a centralized way, where the users′ data and models are stored in the centralized server. In case of an untrustworthy centralized server, the collected sensitive attributes might be leaked by unauthorized usage or data breach. For instance, the personal data of more than half a billion Facebook users was leaked online for free in a hacker forum in 2021[1]. Second, the private information of users can also be leaked from model release or the provided services due to privacy attacks. Taking the online service for brain disease classification as example, membership inference attack can figure out the patients that covered in the training dataset, which severely threaten the privacy of patients. Moreover, various types of privacy attacks[16, 53] such as link inference and attribute inference have been proved effective to steal users′ information from the pretrained model. Therefore, it is crucial to develop privacy-preserving GNNs to achieve trustworthiness.

There are several surveys on the privacy aspect of machine learning model. In [62], it comprehensively reviews the current privacy attacks in i.i.d data. Privacy-preserving methods are reviewed in [63–66] as well. However, they are overwhelmingly dedicated to the privacy issue on models for i.i.d data such as images and text, and rarely discuss the privacy attacks and defense methods on graphs; while these methods are challenged by the topology information in graphs and the message-passing mechanism of GNN models. Therefore, in this section, we give the overview of the privacy attacks on GNNs and privacy-preserving GNNs to defend against privacy attacks. We also include related datasets and the applications of privacy-preserving GNNs followed by future research directions on privacy-preserving GNNs.

3.1 Taxonomy of privacy attacks

In this subsection, we will introduce the categorization of privacy attacks on GNNs according to the target private information. We will also briefly explain two settings of the attacker′s accessible knowledge for conducting privacy attacks. Finally, we will present more details of existing privacy attack methods on GNNs.

3.1.1 Types of privacy attacks on graph neural networks

The goal of privacy attacks on GNNs is to extract information that is not intended to be shared. The target information can be about the training graph such as the membership, sensitive attributes of nodes, and connections of nodes. In addition, some attackers aim to extract

---

[1] https://www.bbc.com/news/technology-56745734

the model parameters of GNNs. Based on the target knowledge, the privacy attacks can generally be split into four categories:

- **Membership inference attack**: In membership inference attack, the attackers try to determine whether a target sample is part of the training set. For example, suppose researchers train a GNN model on social network of flu patients to analyze the propagation of virus. The membership inference attack can identify if a target subject is in training patient network, resulting in information leakage of the subject. Different from i.i.d data, the format of the target samples can be nodes or graphs. For instance, for node classification task, the target samples can be subgraph of the target node's local graph[54] or only contain the node attributes[55]. For graph classification task, the target sample is a graph to be classified[56].

- **Reconstruction attack**: Reconstruction attack, also known as model inversion attack, aims to infer the private information of the input graph. Since the graph-structure data is composed of graph topology and node attributes, the reconstruction attack on GNNs can be split into structure reconstruction, i.e., infer the structures of target samples, and attribute reconstruction (also known as attribute inference attack), i.e., infer the attributes of target samples. Generally, the embeddings of the target samples are required to conduct the reconstruction attack.

- **Property inference attack**: Different from attribute reconstruction attack, property inference attack aims to infer dataset properties that are not encoded as features. For instance, one may want to infer the ratio of women and men in a social network, where this information is not contained in node attributes. The attacker may also be interested in structure-related properties such as degrees of a node, which is the number of friends of the target user in a social network[16].

- **Model extraction attack**: This attack aims to extract the target model information by learning a model that behaves similarly to the target model. It may focus on different aspects of the model information, which results in two goals in model extraction: 1) The attacker aims to obtain a model that matches the accuracy of the target model; 2) The attacker tries to replicate the decision boundary of the target model. Model extraction attack can threaten the security of model for API service[67] and can be a stepping stone for various privacy attacks and adversarial attacks.

Table 2 categorizes existing methods on privacy attacks on GNNs based on the attack types. We will introduce the details of these methods in Section 3.2.

3.1.2 Threat models of privacy attacks

To conduct privacy attacks, auxiliary knowledge about the target GNN and/or dataset is usually possessed by attackers. In this subsection, we introduce the categorization of threat models of privacy attacks in the aspect of attackers' knowledge. Generally, based on whether the model parameters of the target GNN are available, attacker's knowledge about the threat model can be split into two settings, i.e., white-box attack and black-box attack:

- **White-box attack**: In white-box attack, the model parameters or the gradients during training is accessible for the attackers. Apart from the knowledge about the trained GNNs, the attacks may require some other knowledge such as the nodes/graphs to be attacked in inference attacks and a shadow dataset, i.e., dataset that follows the same distribution as the training dataset of the target GNN. The white-box attack can be used to attack pretrained GNNs whose model is publicly released[68]. It is also practical during the training process of federated learning[53] where the intermediate computations.

- **Black-box attack**: In contrast to white-box attack, the parameters of the target GNN are unknown in black-box attack; while the architecture of the target GNN and hyperparameters during training may be known. In this setting, attackers are generally allowed to query the target GNN model to get the prediction vectors or embeddings of the queried samples. Similar to the white-box attacks, shadow datasets and the target nodes/graphs are also required to conduct black-box attacks. A practical example of the black-box privacy attack is to attack the API service that sends the output of the GNN models when receiving queries from the users.

The categorization of existing privacy attack methods according to the assumption on attacker's knowledge is shown in Table 3.

Table 2　Different types of privacy attack methods on GNNs

Privacy attack types	References
Membership inference	[53], [54], [55], [56]
Property inference	[16]
Reconstruction attack	[15], [16], [53], [57]
Model extraction	[58], [59]

Table 3　Categorization of attacker's knowledge

Knowledge	References
White-box	[53], [57]
Black-box	[53], [54], [55], [56], [58], [59], [16], [15]

3.2 Methods of privacy attack on graph neural networks

The unified framework. Supervised privacy attack is a common design strategy of privacy attacks[15, 16, 53–56, 58, 59]. The core idea of supervised privacy attack methods is to utilize the shadow dataset and the output of the

target models to get the supervision for training a privacy attack mode, which can be described as a unified framework shown in Fig. 2(a). As shown in Fig. 2(a), the attacker uses a shadow dataset $\mathcal{D}_S$ as input to the target model f_T to obtain the predictions or embeddings. Then, ground-truth of various types of privacy attacks on the shadow dataset can be attained. With the attack labels from the shadow dataset, attackers can train an attack model that performs inference based on the outputs of the target models by:

$$\min_{\theta_A} \frac{1}{|\mathcal{D}_S|} \sum_{\mathcal{G}_i \in \mathcal{D}_S} l(f_A(f_T(\mathcal{G}_i)), y_i) \tag{4}$$

where $\mathcal{G}_i$ indicates the samples from the shadow dataset, which can be subgraphs of node v_i's local graph for node classification or a sample graph for graph classification. y_i represents the extracted attack labels of the sample $\mathcal{G}_i$. As illustrated in Fig. 2(a), it can be varied from attributes to network properties for different privacy attacks. $l(\cdot)$ denotes the loss function such as cross entropy loss to train the attack model f_A. θ_A denotes the parameters of the attack model f_A. After the attack model is trained, privacy attack on the target example $\mathcal{G}_t$ can be conducted by $f_A(f_T(\mathcal{G}_t))$. Next, we will give more details about the methods of each types of privacy attacks.

Membership inference attack. The membership inference attack aims to identify if a target sample was used for training the target model f_T. The privacy leakage of membership is caused by the overfitting of the model on the training dataset, which leads to the prediction vectors (predicted label distributions) on training set and target set follow different distributions[56, 69]. Thus, an attacker can utilize the prediction vector to judge if a data instance was in the training set for f_T. To learn a membership inference attack model, the most common way is to apply shadow training to obtain supervision for membership inference and train an attack model. The process of shadow training is shown in Fig. 2(b). In shadow training, part of the shadow dataset $\mathcal{D}_S^{train}$ are used

to train a surrogate model f_S to mimic the behaviors of the target model as

$$\min_{\theta_S} \frac{1}{|\mathcal{D}_S^{train}|} \sum_{\mathcal{G}_i \in \mathcal{D}_S^{train}} l(f_S(\mathcal{G}_i)f_T(\mathcal{G}_i)) \tag{5}$$

where $\mathcal{G}_i$ is a graph for graph classification or k-hop subgraph centered at node v_i for node classification. $f_T(\mathcal{G}_i)$ denotes the predicted label distribution of $\mathcal{G}_i$ and $l(\cdot)$ is loss function such as cross entropy loss to ensure $f_S(\mathcal{G}_i)$ is similar to $f_T(\mathcal{G}_i)$. Since $\mathcal{G}_i \in \mathcal{D}_S^{train}$ is used for training f_S, the predictive probability vectors of $\mathcal{G}_i$ from f_S, i.e., $[f_S(\mathcal{G}_i), y_i]$, can be labeled as positive, where y_i is the ground-truth class label of $\mathcal{G}_i$, used to help the attack model to judge if $f_S(\mathcal{G}_i)$ is overfitting. Similarly, the probability vectors of other shadow samples $\mathcal{D}_S^{Out} = \mathcal{D}_S - \mathcal{D}_S^{train}$ can be labeled as negative samples for membership inference attack. Then, the prediction vectors and the obtained labels for membership inference will be used to train an attacker model such as logistic regression, which can infer whether a sample is in the training dataset of the target model or not.

Membership inference attack has already been extensively investigated on i.i.d data[69–71]. Due to the great success of GNNs, recently, membership inference attack on graphs has raised increasing attention[53–56], which generally follow the same scheme of shadow training. In [54], membership inference based on subgraphs of the target node's local graph is investigated for node classification task. More specifically, the shadow dataset $\mathcal{D}_S$ in [54] is a graph that comes from the same underlying distribution as the graph used for training. A GCN is deployed as the shadow model f_S. He et al.[55] further investigate the attack on target samples that are only provided with node features. To infer the membership of a singe test node, the attacker model is trained to distinguish $\{f_S(\boldsymbol{x}_i) : v_i \in \mathcal{D}_S^{train}\}$ and $\{f_S(\boldsymbol{x}_i) : v_i \in \mathcal{D}_S^{Out}\}$, where $f_S(\boldsymbol{x}_i)$ denotes the prediction of f_S when only attributes of a single node v_i is fed into the surrogate GNN. Wu et al.[56] also adopt the introduced framework. The major differ-

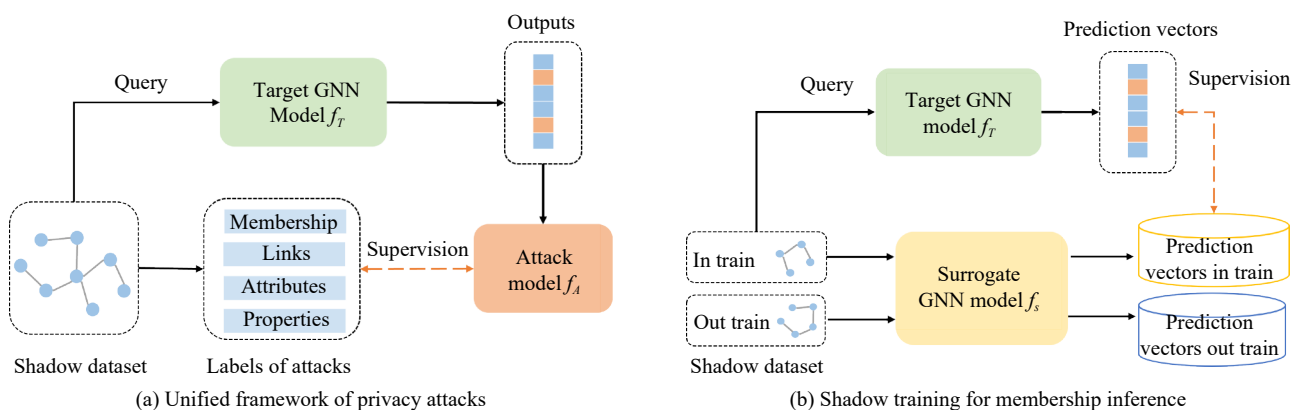(a) Unified framework of privacy attacks   (b) Shadow training for membership inference

Fig. 2   The illustration of the privacy attack methods.

ence is that they achieve the membership inference attack on graph classification task.

Reconstruction Attack. As mentioned in Section 3.1.1, reconstruction attack aims to infer the sensitive attributes or/and links in target datasets. Due to the message passing of GNNs, the learned node/graph embeddings capture both node attributes and graph structure information. Thus, existing reconstruction attack method[53] reconstructs the information from node embeddings $\boldsymbol{H} = [\boldsymbol{h}_1, \cdots, \boldsymbol{h}_N]$ learned by the target GNN. For attribute reconstruction, f_A can simply be a multilayer perceptron (MLP) and reconstruct the attributes as $\hat{\boldsymbol{X}} = MLP(\boldsymbol{H})$. For link inference, f_A generally predicts the link based on the embeddings of node v_i and v_j by $w(i,j) = MLP(\boldsymbol{h}_i, \boldsymbol{h}_j)$. Following the unified framework, a shadow graph $\mathcal{G}_S$ is used to provide the adjacency matrix $\boldsymbol{A}_S$ and sensitive attributes $\boldsymbol{X}_S$ as supervision. The node embeddings $\boldsymbol{H}_S$ of the shadow graph are assumed to be available. Then, the attack model can be trained in a supervised manner. For attribute reconstruction attack, the training loss is given as $\min_{\theta_A} \sum_{v_i \in \mathcal{G}_S} l_{attr}(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i)$, where l_{attr} could be MSE loss for continuous attributes and cross entropy loss for categorical attributes. As for the link inference, the objective function is $\min_{\theta_A} \|\hat{\boldsymbol{A}}_S - \boldsymbol{A}_S\|_F^2$, where $\hat{\boldsymbol{A}}_S$ is the adjacency matrix reconstruted by f_A. Similar to link prediction, negative sampling may also be applied here. Reconstructing adjacency matrix on graph embeddings are also investigated in [16]. Different from [53], the attacker model in [16] directly infers the adjacency matrix with the graph embedding by $\hat{\boldsymbol{A}} = MLP(\boldsymbol{h}^G)$, where $\boldsymbol{h}^G$ denotes a graph embedding.

For some applications, the embeddings might not be available while the prediction vector of an instance can be obtained by querying the target model. Therefore, He et al.[15] propose to infer the link of two nodes with their prediction vectors as $w(i,j) = MLP(\boldsymbol{y}_i, \boldsymbol{y}_j)$, where $\boldsymbol{y}_i$ is the prediction vector of node v_i from target model f_T. Following shadow training, the authors firstly query the prediction vectors of shadow dataset from the target model f_T. Then, a surrogate model is trained on the shadow dataset and queried prediction vectors. Finally, the link inference attacker can be learned with the supervision generated from the surrogate model.

The aforementioned methods all focus on black-box settings. However, with the development of pretraining GNN and federated learning that share the model parameters, white-box reconstruction attack methods also start to attract attentions. For example, GraphMI[57] proposes to reconstruct the adjacency matrix in a white-box setting where the trained model parameters are known. Intuitively, the reconstructed adjacency matrix $\boldsymbol{A}$ will be similar to the original adjacency matrix if the loss between true node label y_i and predictions using the reconstructed matrix, $f_\theta(\boldsymbol{X}, \boldsymbol{X})_i$ is minimized. In addition, the graph structure is updated to ensure accurate predic-

tions from GNN model under the feature smoothness constraint. Formally, GraphMI aims to solve the following optimization problem:

$$\min_{\boldsymbol{A} \in \{0,1\}^{N \times N}} \frac{1}{|\mathcal{V}_L|} \sum_{v_i \in \mathcal{V}_L} l(f_\theta(\boldsymbol{A}, \boldsymbol{X})_i, y_i) +$$
$$\alpha \mathrm{tr}(\boldsymbol{X}^\mathrm{T} \boldsymbol{L} \boldsymbol{X}) + \beta \|\boldsymbol{A}\|_1 \qquad (6)$$

where $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$ is the Laplacian matrix of $\boldsymbol{A}$ and $\boldsymbol{D}$ is the diagonal matrix of $\boldsymbol{A}$. $\|\boldsymbol{A}\|_1$ is the ℓ_1-norm to make $\boldsymbol{A}$ sparse. α and β control the contributions of feature smoothness constraint and sparsity constraint on $\boldsymbol{A}$.

Property inference attack. The property inference attack is still in an early stage. An initial effort is taken in [16] to infer the properties of a target graph by its embedding. The proposed method also follows the unified framework. Let p_i denote the property of a graph $\mathcal{G}_i$ in the shadow dataset $\mathcal{D}_S$, the attack model is trained by $\min_{\theta_A} \frac{1}{|\mathcal{D}_S|} \sum_{\mathcal{G}_i \in \mathcal{D}_S} l(f_A(\boldsymbol{h}_i^G), p_i)$, where $\boldsymbol{h}_i^G = f_T(\mathcal{G}_i)$ is the embedding of $\mathcal{G}_i$. $l(\cdot)$ can be MSE loss or cross entropy loss for different types of properties.

Model extraction attack. The model extraction attack aims to learn a surrogate model that behaves similarly to the target model. The process of training a surrogate model is also included in the membership inference attack, which is shown in Fig. 2(b). Generally, the attacker will first query the target model to obtain predictions on the shadow dataset. It then leverages the shadow dataset and the corresponding predictions to train the surrogate model for model extraction attack, which has been formulated in (5). Following this framework, recent works[58, 59] have investigated the GNN model extraction attacks with different levels of knowledge on shadow dataset and training graph of the target model. For example, the general framework of model extraction is applied for training on the shadow dataset that provided with graph structures[58, 59]. If no structure is given in the shadow dataset, the missing graph structures can be firstly learned. For example, in [59], the graph structure is firstly initialized by KNN on the node attributes then updated by a graph structure learning framework[72].

3.3 Privacy-preserving graph neural networks

As GNNs are vulnerable to privacy attacks and may leak the private information of users, privacy-preserving GNNs are developed to protect the privacy. Current privacy-preserving GNNs generally fall into the following categories, i.e., differential privacy, federated learning, machine unlearning, adversarial privacy-preserving and model ownership verification. The categorization of the privacy-preserving GNNs are listed in Table 4. Next, we will introduce representative and state-of-the-art methods in each category.

Table 4   The categorization of privacy-preserving graph neural network

Category	Privacy attacks to defense	References
Differential privacy (DP)	Membership inference	[73], [74], [75], [76]
Federated learning	–	[77], [78], [79], [80], [81]
Federated learning + DP	Membership inference	[34], [82], [83]
Machine unlearning	–	[84], [85], [86], [87]
Adversarial privacy-preserving	Attribute reconstruction	[88], [33]
	Structure reconstruction	[89]
Model ownership verification	Model extraction	[90], [91], [92]
Other privacy protection methods	Membership inference	[93], [54]

3.3.1 Differential privacy for privacy-preserving GNNs

Differential privacy (DP)[94] is a popular approach that can provide privacy guarantee of training data. The core idea of differential privacy is that if two datasets differ only by one record and are used by the same algorithm, the outputs of the algorithm on the two datasets should be similar. With differential privacy, the impact of a single sample is strictly controlled. Thus, the membership inference attack can be defended by DP with theoretical guarantee. Formally, differential privacy is defined as follows:

Definition 1 ((ϵ, δ)-differential privacy[94]). Given $\epsilon > 0$ and $\delta \geq 0$, a randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ) differential privacy, if for any adjacent datasets D and $D' \in \mathcal{R}$ and for any subsets of outputs $\mathcal{S}$, the following equation is met:

$$P(\mathcal{M}(D) \in \mathcal{S}) \leq e^{\epsilon} P(\mathcal{M}(D') \in \mathcal{S}) + \delta \qquad (7)$$

where ϵ is the privacy budget to trade-off the utility and privacy. A larger ϵ will lead to stronger privacy guarantee but weaker utility. When $\delta = 0$, it is equivalent to ϵ-differential privacy. (ϵ, δ)-DP allows for the possibility that plain ϵ-DP is broken with a small probability δ.

To achieve (ϵ, δ)-differential privacy, some additive noise mechanisms such as Gaussian mechanism[95] and Laplace mechanism[94] are widely adopted. Based on the privacy budget and the mechanism to be protected, certain levels of Gaussian noise or Laplace noise will be injected to achieve a differentially private mechanism. Recently, various differential-privacy preserving deep learning methods[74, 96–98] are proposed to protect the training data privacy. For instance, NoisySGD[96] adds noises to the gradients during model training so the trained model parameters will not leak training data with certain guarantee. PATE[98] firstly trains an ensemble of teacher models on subsets of sensitive training data that are split disjointly. Then, the student model will be trained with the aggregated output of the ensemble on public data. As a result, the student model is unlikely to be affected by a change of a single sensitive data, which meets the differential privacy. Theoretical guarantee of PATE in privacy

is also analyzed in [98]. In differential privacy, a trusted curator will be required to apply calibrated noise to produce DP. To handle the situation of untrusted curator, local differential privacy methods[74, 97] that perturb users′ data locally before uploading to the central server are also investigated for privacy protection.

To protect the privacy of graph-structured data, many differential privacy preserving network embedding methods are investigated[73–76]. For instance, DPNE[75] applies perturbations on the objective function of learning network embeddings. In [99], a perturbed gradient descent method that guarantees the privacy of graph embeddings learned by matrix factorization is proposed. More recently, several works[73, 74] that focus on differentially private GNNs are explored. For example, the locally private GNN[74] adopts local differential privacy[100] to protect the privacy of node features by perturbing the user′s features locally. Furthermore, a robust graph convolution layer is investigated to reduce the negative effects of the injected noises. PrivGnn[73] extends the private aggregation of teacher ensembles (PATE)[98] for graph-structured data to release GNNs with differential privacy guarantees. In particularly, random subsampling on the training set of teacher and noisy labeling mechanism on the public data are used in PrivGNN[73] to achieve practical privacy guarantees. As for [76], both the user feature perturbation at input stage and loss perturbation at optimization stage are investigated to achieve a privacy-preserving GNN for recommendation. In addition, they also empirically show that their proposed method can help defend against the attribute inference attack as the noises added to node features can prevent the attacker from reconstructing the original sensitive attribute.

3.3.2 Other methods for membership privacy preserving

Beyond differential privacy approaches, recent developments have given rise to a variety of other methods aimed at preserving membership privacy. Next, we will delve into a detailed exploration of these newly established works.

LBP and **NSD**[54]. These two methods are preliminary explorations in defending against membership inference attacks. Specifically, LBP is an output perturbation

method by where noise is infused to the posterior before it is released to end users. Intuitively, noise can obfuscate the posteriors, making it challenging to discern between member and non-member node posteriors. As for NSD, it randomly chooses neighbors of the queried node during inference. This can limit the amount of information used in the target model, thereby protecting membership privacy.

RM-GIB[93]. This work represents a pioneering effort to craft a unified graph neural network framework designed to both preserve membership privacy and defend against adversarial attacks. Dai et al.[93] first connect the information bottleneck (IB) with the membership privacy preservation. In particularly, the proposed RM-GIB principle can be written as

$$\min_{\theta} -I(\boldsymbol{z}_x, \mathcal{N}_S; y) + \beta I(\boldsymbol{z}_x, \mathcal{N}_S; \boldsymbol{x}, \mathcal{N}) \tag{8}$$

where $\boldsymbol{z}_x$ represents the attribute bottleneck encoding the node attribute information, $\mathcal{N}_S$ is a subset of v's neighbors that bottleneck the neighborhood information, and $I(;)$ denotes the mutual information. As analyzed in [93], the regularization in IB (latter term in (8)) can constrain the mutual information between representations and labels on the training set. This constraint can narrow the gap between training and test sets to avoid membership privacy leakage. In addition, RM-GIB collects pseudo labels on unlabeled nodes and integrate them with the given labels in the optimization, further benefiting the membership privacy. The proposed RM-GIB also benefit the robustness, as an attribute bottleneck and a neighbor bottleneck are deployed to remove the redundant information and/or adversarial perturbations in both node attributes and graph topology.

3.3.3 Federated learning for privacy-preserving GNNs

Currently, the majority of deep learning methods require a centralized storage of user data for training. However, this can be unrealistic due to the privacy issue. For example, when several companies or hospitals want to combine their data to train a GNN model, the data of their users are not allowed to be shared according to the privacy terms. Furthermore, the users may be not willing to upload their data to the platform server due to the concern of information leakage. For such situations, the data will remain in the local devices of users or the data holder organizations. To address this problem, federated learning[101, 102] is proposed to collectively learn models with decentralized user data in a privacy-preserving manner. In particular, it aims to optimize the following objective function:

$$\min_{w} \sum_{k=1}^{n} p_k \mathcal{L}_k(\mathcal{D}_k, w) \tag{9}$$

where n is the total number of devices/clients, $\mathcal{D}_k$ is the local dataset stored in the k-th client, and $\mathcal{L}_k$ is the local objective function for the k-th device. The impact of each device is controlled by p_k with $p_k \geq 0$ and $\sum_k p_k = 1$. The p_k is often set as $\frac{1}{n}$ or $\frac{|\mathcal{D}_k|}{|\mathcal{D}|}$, where $|\mathcal{D}| = \sum_k |\mathcal{D}_k|$ is the total size of samples. A general framework of federated learning to solve (9) is given in Fig. 3, where the user data and a local model $f_k(w_t^k)$ are maintained locally in the k-th client in federated learning. In the training step t, each client will compute the local model updates based on their own data. Then, the central server will aggregate the model updates from clients and update the global model parameters to w_{t+1}. The updated global model will be distributed to the clients for future iterations. The federated learning framework is firstly proposed in FedAvg[102], which is the most commonly used federated learning algorithm now. specifically, FedAvg aggregates the model parameters by averaging the updated model parameters from clients as $w_{t+1} = \sum_{k=1}^{n} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} w_t^k$, where w_t^k denotes the updated parameters in the k-th client at step t. More comprehensive survey about federated machine learning can be found in [103].

To protect the user privacy, federated learning has been extended to train GNNs[34, 77–83]. To handle the chal-
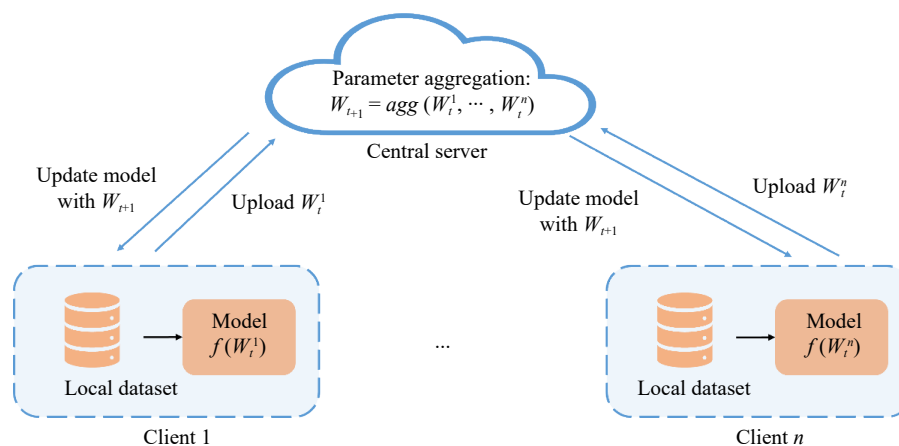

Fig. 3   The illustration of the federated learning

lenges caused by non-i.i.d. graphs, Xie et al.[79] propose to dynamically cluster local systems based on GNN gradients, which can reduce the structure and feature heterogeneity among graphs. In [80], the work proposes a hybrid of meta learning and federated framework. They view the training on a client as a task in meta-learning and learn a global model to mitigate the issue of non-i.i.d data. Then, federated learning methods are leveraged to further update the global model. In [82], vertically federated learning that assumes different clients hold different features and neighborhood information of the user data is proposed. To protect the private data, i.e., node attributes, edges and labels, the computations on private data are carried out by the data-holders. A semi-honest server focuses on computations on encoded node representations that are non-private. Apart from federated learning GNNs in node/graph classification, federated frameworks on GNNs for recommendation are also developed[34, 83]. FedGNN[34] incorporates the high-order user-item interactions by building the local user-item graphs in a privacy-preserving way. Furthermore, noises are injected locally in federated learning to meet the local differential privacy for privacy projection. FeSoG[83] further extends the federated GNNs for social recommendation which involves social information for predictions.

Moreover, decentralized federated learning for GNNs is also explored in [77, 78], where a client can communicate with a set of neighbor clients for aggregation without the central server. Therefore, existing algorithms[77, 78] propose to aggregate the local model with neighbor local models. For example, decentralized periodic averaging SGD[77] applies SGD on each client locally and synchronizes parameters with their neighbors every certain number of iterations.

3.3.4 Machine unlearning on graph-structured data

One of very important regulations on privacy is to ensure the right to be forgotten[20]. This principle demands not only the deletion of data from storage but also the removal of related information from trained AI models. Retraining can be a possible solution for training data removal. However, this can be overwhelmingly expensive especially for models trained on large-scale datasets. Therefore, machine unlearning methods[104] have been developed to efficiently erase the user information in deep models including graph neural networks. Based on the targets, machine unlearning can be split into exact unlearning and approximate unlearning.

Definition 2 (Exact unlearning[104]). Given a learning algorithm $A(\cdot)$, a dataset $\mathcal{D}$, and a forget set $\mathcal{D}_f \subset \mathcal{D}$, the exact unlearning process $U(\cdot)$ is required to meet:

$$P(A(\mathcal{D}\backslash\mathcal{D}_f)) = P(U(\mathcal{D}, \mathcal{D}_f, A(\mathcal{D}))) \qquad (10)$$

where $P(A(\mathcal{D}))$ denotes the distributions of all models trained on $\mathcal{D}$ by the algorithm $A(\cdot)$.

SISA (Sharded, Isolated, Sliced and Aggregated)[104] is

one popular framework for exact unlearning. In this approach, the training set $\mathcal{D}$ will be first partitioned into K disjointed shards. Subsequently, K independent models are trained with the K shards. Predictions are given by assembling the outputs from the these models. When a data point x requests removal, only the model using the shard containing x will be retrained on the small shard. For the graph-structured data, the partition will destroy the training graph's structure which largely degrades the utility. Therefore, GraphEraser[84] introduces a balanced graph partition which splits graphs based on community/clusters.

Definition 3 ((ϵ, δ)-approximate unlearning[105]). Given $\epsilon > 0$ and $\delta \geq 0$, a unlearn process $\mathcal{M}$ satisfies (ϵ, δ) approximate unlearning, if for any data point $z \in \mathcal{D}$ and model sets $\mathcal{T}$, the following equation is met:

$$P(U(\mathcal{D}, \mathcal{D}_f, A(\mathcal{D})) \in \mathcal{T}) \leq e^\epsilon P(A(\mathcal{D}\backslash\mathcal{D}_f) \in \mathcal{T}) + \delta \quad (11)$$

where ϵ is the budget to trade-off the utility and privacy. A larger ϵ will lead to stronger privacy guarantee but weaker utility. (ϵ, δ)-approximate unlearning allows that plain ϵ-approximate unlearning is broken with a small probability δ. Existing methods[85–87] mainly adopt the influenced function-based parameter updating for approximate unlearning by $\theta^- = \theta + H_\theta^{-1}\Delta$, where $H_\theta^{-1}\Delta$ is the influence function of the training point to be unlearned. This is feasible because influence functions quantify how model parameters will adjust if the loss of a particular data point is excluded from training. However, merely adopting influence functions does not assure approximate unlearning because gradient residual persists after updating model parameters. Certain noises in the training would be necessary to ensure the gradient residual would not leak privacy.

3.3.5 Adversarial privacy-preserving graph neural networks

To defend against the sensitive attributes/links leakage attack, adversarial learning is adopted for privacy-preserving GNNs[33, 88, 89]. Let $\boldsymbol{H}$ be node representations learned by the encoder/GNN as $\boldsymbol{H} = f_E(\mathcal{G}; \theta)$. The core idea of the adversarial privacy-preserving is to adopt an adversary f_A to infer sensitive attributes from node representations $\boldsymbol{H}$ while the encoder f_E aims to learn representation that can fool f_A, i.e., making f_A unable to infer sensitive attributes. It is theoretically shown in [89] that through this minmax game, the mutual information between learned representation and sensitive attribute, $MI(\boldsymbol{H}, s)$, can be minimized, which protects the sensitive information from leakage. The process can be formally written as

$$\min_{\theta_E} \max_{\theta_A} \mathcal{L}_{utility}(f_E(\mathcal{G}; \theta_E)) - \beta \mathcal{L}_{adv}(f_A(\boldsymbol{H}; \theta_A)) \quad (12)$$

where θ_E and θ_A are parameters of encoder f_E and

adversary f_A, respectively. $\mathcal{L}_{utility}$ is the loss function to ensure the utility of the learned representations such as classification loss and reconstruction loss. $\mathcal{L}_{adv}$ is the adversarial loss which generally is cross entropy loss of sensitive attribute prediction of the adversary based on node representations $\boldsymbol{H}$. β is the hyperparameter to balance the contributions of these two loss terms.

Adversarial privacy-preserving GNN is firstly proposed in [88] to defend against attribute inference attack, where link prediction loss and the node attribute prediction loss are combined together for utility. Let $\boldsymbol{x}_v^c$ denotes the c-th attribute of node v and $f_c(\boldsymbol{h}_v)$ denotes the prediction of the c-th attribute based on the representation of node v. The utility loss can be written as

$$\mathcal{L}_{utility} = \sum_{i=1}^{N}\sum_{j=1}^{N} l_{edge}(\boldsymbol{A}_{ij}, \hat{\boldsymbol{A}}_{ij}) + \\ \alpha \sum_{v\in\mathcal{V}}\sum_{c\in\mathcal{C}} l_{attr}(\boldsymbol{x}_v^c, f_c(\boldsymbol{h}_v)) \qquad (13)$$

where l_{edge} and l_{attr} denote loss functions for link prediction and attribute reconstruction, $\mathcal{C}$ denotes the set of attributes to be reconstructed. As for the adversarial loss, it is $\mathcal{L}_{Adversarial} = \sum_{v\in\mathcal{V}_S} -[s_v\log(\hat{s}_v) + (1 - s_v)\log(1-\hat{s}_v)]$, where $\hat{s}_v = f_A(\boldsymbol{h}_v)$ and $\mathcal{V}_S$ is the set of nodes with sensitive attributes. GAL[33] deploys a WGAN-based adversarial privacy-preserving for information obfuscation of sensitive attributes. Both attribute privacy protection and link privacy protection with adversarial learning are discussed in [89].

3.3.6 Model ownership verification

Nowadays, it can be overwhelmingly expensive to train a high-performance model due to the demands in high-quality data collection and expensive computation cost. For example, the well-known ChatGPT is reported to cost around 12 million dollars to train once. Similarly, models for graph-structured data, such as graph contrastive learning on molecules[106], also tend to require massive computation on large-scale data. Therefore, trained DNNs is considered as deep intellectual property (IP) with high business values. It is necessary to protect the deep models from stealing and abusing by adversaries. Extensive model ownership verification methods have been proposed for deep neural networks for images and text[107]. Recently, model ownership verification methods for graph neural networks are emerging[90–92]. In this subsection, we will illustrate the general workflow of model ownership verification followed by recent advances in protecting GNNs.

Workflow of model ownership verification. As it is shown in Fig. 4, the workflow protecting the deep IP mainly consists of two stages, i.e., IP construction and IP verification.

● In the IP construction phase, an IP identifier for the deep model will be built either in a invasive or non-invas-
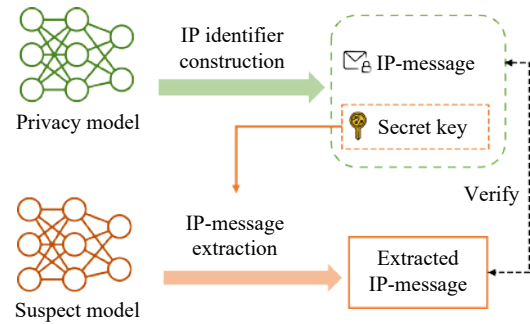

Fig. 4   An illustration of the workflow of model ownership verification

ive way. Generally, the IP identifier is in the form of key-message pair. Model owners possess the secret keys, which can be predefined matrices or special input samples. The IP messages could be a bit string or model outputs that are triggered by the secret keys.

● In the verification phase, given the secret keys, we will testify whether the same IP identifier exists in the suspect model. Specifically, a stealing model will convey the same IP message. On the contrary, an independent model will not output the IP message.

Based on whether the IP identifier is constructed in an invasive way, model ownership verification for GNNs can be split into watermarking methods[90, 91] and fingerprinting methods[92].

GNN watermarking. It is an invasive solution that will embed the detectable and unforgettable IP identifier to the GNN model to obtain a watermarked GNN. Current methods[90, 91] focus on applying backdoor attacks to obtain watermarked GNN model. Specifically, backdoor attacks aim to learn a model that predicts the target class y_t given arbitrary sample attached with the predefined trigger $\mathcal{G}_t$. Therefore, the trigger graph $\mathcal{G}_t$ can work as the secret key for watermarking. The target class y_t will be the IP message. The training process in [90, 91] is the same as the graph backdoor attacks. Firstly, a trigger graph $\mathcal{G}_t$ will be generated. Then, samples with the defined trigger $\mathcal{G}_t$ will be labeled as a target class y_t and join the training process of GNN. More details about the backdoor attacks are illustrated in Section 4.2.5. As for watermarking GNN model without using backdoor attacks, it still remains further investigation.

GNN fingerprinting. Fingerprinting is a noninvasive solution which aims to build IB identifier for a trained model. The main idea is based on the assumption that a trained neural network models will exhibit distinct characteristics when compared to an independently trained model. These differences can be model predictions, decision boundaries, adversarial samples, etc. Fingerprinting techniques have been deeply studied in the context of i.i.d data[107]. However, for GNN fingerprinting, there is only one initial effort named GROVE[92], which proposes a fingerprinting scheme based on GNN embeddings. More precisely, a set of test samples, denoted as $\mathcal{D}_f$, serve as

the secret key for fingerprinting. During the verification, the similarity between the embeddings from protected model and suspect model will be computed as the IP message. If the similarity of embeddings from the protected model and suspect model are high, it indicates a high probability of model stealing.

3.4 Datasets for privacy-preserving graph neural networks

In this subsection, we list the datasets that have been used in the literatures about GNN′s privacy. The statistics of the datasets along with papers used the datasets are presented in Table 5.

• **Cora, Citeseer, PubMed, DBLP**[108, 109]: Cora, Citeseer, PubMed, and DBLP are citation network datasets. Cora consists of seven classes of machine learning papers. CiteSeer has six classes. Papers are represented by nodes, while citations between two papers are represented by edges. Each node has features defined by the words that appears in the paper′s abstract. Similarly, PubMed is a collection of abstracts from three types of medical papers. The data of DBLP comes from four research areas.

• **Facebook, LastFM**[110]: Facebook and LastFM are social network datasets. Different user accounts are represented by nodes that are connected by edges. Each user node in Facebook has features such as gender, education, hometown, and location. LastFM was collected through the following relationship in social network.

• **Flickr**[111]: Flickr is an online photo management and sharing application. The edges reflect the common properties shared between two images, whereas the nodes

represent an image submitted to Flickr. Users′ node features are specified by their tags, which indicate their interests.

• **Reddit**[3]: The Reddit dataset represents the post-to-post interactions of a user. An edge between two posts indicates that the same user commented on both posts. The labels correspond to the community that a post is associated with.

• **Coauthor**[112]: Coauthor is a dataset of co-authorship. Nodes represent authors, and edges connect two authors if they co-authored a paper. Features are collected by keywords in the author′s papers. The label of a node is the area that the author focuses on.

• **ACM**[113]: In ACM, nodes are papers. Edges mean if there are same authors in two papers. Features of a node are keywords of the paper, while labels are the conferences that the papers are published.

• **PROTEIN, DD, ENZYMES**[114]: DD, ENZYMES, and PROTEINS are macromolecules datasets. Nodes are secondary structural elements labeled with their type and a variety of physical and chemical data. If two nodes are neighbors along the amino acid sequence or one of the three nearest neighbors in space, an edge links them. ENZYMES assigns enzymes into six classes, which reflect the catalyzed chemical reaction. The labels in PROTEINS show whether a protein is an enzyme. In DD, nodes are amino acids, and edges are their spatial proximity.

• **AIDS, NCI1, OVCAR-8H, COX2**[16, 114]: AIDS, NCI1, OVCAR-8H, and COX2 are molecule datasets. Atoms are represented by nodes, while chemical bonds are represented by edges. The node features consist of atom types. The label is decided by toxicity or biological

Table 5   Datasets in privacy

Type	Dataset	#Graphs	Avg.Nodes	Avg.Edges	#Features	References
Citation	Cora	1	2 708	5 429	1 433	[53], [54], [55], [15], [57], [58]
	Citeseer	1	3 312	4 715	3 703	[53], [54], [55], [15], [57], [58], [59]
	PubMed	1	19 717	44 338	500	[53], [54], [15], [58], [59]
	DBLP	1	17 716	105 734	1 639	[59]
Authorship	Coauthor	1	34 493	247 962	8 415	[59]
	ACM	1	3 025	26 256	1 870	[59]
Social networks	Facebook	1	4 039	88 234	–	[53]
	LastFM	1	7 624	27 806	7 842	[53], [55]
	Reddit	1	232 965	57 307 946	602	[54]
Image	Flickr	1	89 250	449 878	500	[54]
Bioinformatics	PROTEINS	1 113	39.06	72.82	29	[56], [15]
	DD	1 178	284.32	715.66	89	[56], [16]
	ENZYMES	600	32.63	62.14	21	[56], [16], [15], [57]
Molecule	NCI1	4 110	29.87	32.30	37	[56], [16]
	AIDS	2 000	15.69	16.20	42	[16], [15], [57]
	OVCAR-8H	4 052	46.67	48.70	65	[56], [16]

activity in drug discovery projects.

3.5 Applications of privacy preserving GNNs

Pretraining and model sharing. Nowadays, there is an increasing trend of pretraining models[106, 130] on large-scale datasets to benefit the downstream tasks. In practice, the pretrained model will often be shared to other parties for their use. However, the pretrained model itself has embedded the information of the training data, which can cause private data leakage by privacy attacks such as membership inference and attribute inference. The privacy-preserving GNNs can be applied to address this concern. For instance, differential privacy-preserving GNNs[73, 74, 76] can be adopted in the pretraining phase to defend against the membership inference attack.

Distributed learning. Due to challenges of processing the large amount of data such as privacy concerns, computational cost, and memory capability, the demand of distributed learning is increasing dramatically. In this situation, federated learning on GNNs provide solutions to distributed learning by processing data in local devices. In addition, combining the other privacy-preserving methods such as differential privacy[34], the client′s data can be protected from privacy attacks.

Healthcare. Graph-structured data such as protein molecules, brain network, and patient network are pervasive in healthcare domain. GNNs have been trained on these private healthcare data for various applications. For example, GNNs have been used to process electronic health record (EHR) of patients for diagnosis prediction[60]. GNNs are also deployed to better capture the graph signal of brain activity for medical analysis[131]. To ensure the privacy of sensitive data of patients, privacy-preserving GNNs are required for the applications in healthcare domain.

Recommendation system. GNNs are widely applied in recommendation system to involve social context and better utilization of high-order neighbor information[34]. Similarly, information can be leaked from the GNN-based recommendation system. To protect the user privacy, various privacy-preserving recommendation systems have been proposed[34, 76, 83].

3.6 Future research directions of privacy preserving GNNs

Defense against various privacy attacks. Though many privacy-preserving GNNs have been proposed, they mostly focus on defending against membership inference attack and attribute reconstruction attack. The privacy-preservation GNNs against structure attack, property inference attack, and model extraction attack are less studied. Therefore, it is promising to develop privacy-preserving GNNs against various privacy attacks.

Privacy attack and preservation in GNN pre-

training. Model pretraining have been a common scheme to benefit the downstream tasks that are lack of labels. Recently, pretraining of GNNs with supervised tasks[132] and self-supervised tasks[106, 130] have achieved great success. The parameters of pretrained GNNs will be released for downstream tasks, which may lead to private information leakage. However, existing privacy attacks mostly focus on black-box settings and do not investigate the information leakage caused by model releasing. Hence, privacy attack and the corresponding defense methods for pretrained GNN modes need to be explored.

Trade off between privacy and utility. Though methods that apply differential privacy, federated learning or adversarial learning have been proposed to protect the privacy of training data, the relations between the privacy protection performance and the prediction accuracy are rarely discussed. For example, in differentially private GNNs[73, 74, 76], the actual performance in defending against various privacy attacks is generally not evaluated. And in adversarial privacy-preserving[33, 88, 89], how to control the balance between the prediction performance and privacy protection is still not well discussed.

4 Robustness of graph neural networks

As an extension of neural networks on graph-structured data, GNNs are also vulnerable to adversarial attacks. In addition, due to the message-passing mechanism and graph structure, GNNs can be negatively affected by adversarial perturbations on both graph structures and node attributes. For example, Nettack[120] can fool GNNs to give false predictions on target nodes by poison the training graph with unnoticeable perturbations to graph structure and node attributes. NIPA[17] manages to significantly reduce the global node classification performance of GNNs by injecting a small amount of labeled fake nodes to the training graph. The vulnerability of GNNs has arisen tremendous concerns on adopting GNNs in safety-critical domains such as credit estimation and healthcare. For example, fraudsters can create several transactions with deliberately chosen high-credit users to escape GNN-based fraud detectors, which can cause tremendous loss to individuals, and institutions. Hence, developing robust GNNs is another important aspect of trustworthiness and many efforts have been taken. There are already several comprehensive surveys about adversarial attacks and defenses on graphs[26, 37, 38, 133]. Therefore, in this section, we briefly give the overview of adversarial learning on graphs, but focus more on methods in emerging directions such as scalable attacks, graph backdoor attacks, and recent defense methods.

4.1 Threat models of graph adversarial attacks

Graph adversarial attacks aim to degrade the performance of GNNs or to make GNN models give desired out-

put by injecting deliberate perturbations to the graph dataset. Generally, attackers are constrained in the knowledge about the data and the model they attack, as well as the capability of manipulations on the graph. In this subsection, we introduce the threat models in various aspects to show different settings of graph adversarial attacks.

Attackers′ knowledge. Similar to privacy attacks, attackers need to possess certain knowledge about the dataset and target model to achieve the adversarial goal. Based on whether the model parameters are known for the attacker, they can be split into white-box and black-box attack:

• **White-box attack**: In this setting, the attacker knows all information about the model parameters and the training graph such as adjacency matrix, attribute matrix and labels[115, 116, 118]. Since it is impractical to obtain all information in the real world, the white-box attack is less practical but often used to show the worst performance of a model under adversarial attacks.

• **Black-box attack**: In black-box attack[117, 120, 123], attackers do not have access to the model′s parameters but can access graph dataset. More specifically, the full/partial of the graph structure and node features could be accessible for attackers. Attackers may be allowed to have labels used for training or query the outputs of the target GNN, which could be used to mimic the predictions of the target model to achieve black-box attack.

Attackers′ capability . In adversarial attacks, adversarial perturbations are added to data samples to mislead the target GNN model to give the output desired by the attacker. According to the stage the attack occurs, the attacks can be divided into poisoning attack and evasion attack:

• **Evasion attack**: The perturbations in evasion attacks[115, 117] are added to the graph in the test stage, where the GNN model parameters have been well trained and cannot be affected by attackers. Depending on whether the attacker knows the model parameters, evasion attack can be further categorized into white-box or black-box evasion attack.

• **Poisoning attack**: In poisoning attacks[17, 120, 121], the training graph is poisoned before GNNs are trained. The GNN model trained on the poisoned dataset will exhibit certain designed behaviors such as misclassifying target nodes or having low overall performance. As poisoning attack happens before model training, it belongs to the black-box attack where the model parameters are unknown. Thus, attackers usually train a surrogate model and poison the graph to reduce the performance of the surrogate model. Due to the transferability of adversarial attack, the poisoned graph can also reduce the performance of the target GNN trained on it.

Currently, many graph mining tasks such as semi-supervised node classification and link prediction are trans-

ductive learning, where the test samples participate in training phase. Therefore, most of existing works focus on poisoning attacks, which are often more practical for graph mining.

Based on the way that the graph data is perturbed, the adversarial attacks can be categorized into manipulation attacks, node injection attack, and backdoor attacks:

• **Manipulation attack**: In manipulation attack, an attacker manipulates either graph structure or node features to achieve the attack goal. For example, Nettack[120] perturbs the graph by deliberately adding/deleting edges and revising the node attributes with a greedy search algorithm based on gradients. To make the perturbation more unnoticeable, ReWatt[119] poisons the training graph by rewiring edges with reinforcement learning.

• **Node injection attack**: Different from manipulation attack that modifies the original graph, node injection attack aims to achieve the adversarial goal by injecting malicious nodes into the graph[17, 122, 125]. Compared to manipulation attack, node injection attack is more practical. For example, in e-commercial network, attackers need to hack servers or user accounts to manipulate the network; while injecting new malicious accounts is much easier.

• **Backdoor attack**: Backdoor attacks[127, 128] inject backboor triggers to the training set to poison the model. The backdoor trigger is a predefined or learned pattern. The attacker relabel training nodes/graphs attached with the backdoor trigger to the target label so that a GNN trained on the poisoned dataset will predict any test sample with backdoor trigger to the target label. Compared with other types of adversarial attacks, backdoor attack on GNNs is still in an early stage.

Attackers′ goal. Based on whether the goal of the attacker is to misclassify a set of target instances or reduce the overall performance, threat models can also be categorized as:

• **Targeted attack**: The attacker aims to fool a GNN model to misclassify a set of target nodes[120, 128]. Meanwhile, the attacker might want the performance of the target model on non-targeted samples remain unchanged to avoid being detected.

• **Untargeted attack**: The untargeted attack[17, 121] aims to reduce the overall performance of the target GNN model. Since evasion attack cannot affect the parameters, it can only be achieved by poisoning the dataset in the training stage.

The categorization of the existing representative graph adversarial attacks in different aspects are summarized in Table 6. Next we will give more details of these methods.

4.2 Graph adversarial attack methods

In this subsection, we first give a unified formulation of adversarial attacks followed by representative methods

Table 6   Categorization of representative graph adversarial attacks

Aspect	Category	References
Knowledge	White-box	[30], [115], [116], [117], [118]
	Black-box	[117], [119], [120], [116], [121], [115], [122], [17], [123], [124]
Capability	Evasion attack	[30], [115], [116], [117], [125], [119], [126], [119], [118]
	Poisoning attack	[120], [121], [17], [123], [124], [122], [127], [128], [129], [118]
Attackers' goal	Targeted attack	[117], [122], [125], [127], [128], [120], [119], [116], [129], [123]
	Untargeted attack	[17].[121], [115], [118], [123], [30], [126], [124]

in evasion attacks and poisoning attacks. Finally, we survey recent advances in backdoor attacks and scalable attacks on GNNs.

4.2.1 A unified formulation of adversarial attack

The adversarial attacks can be conducted on both node-level and graph-level tasks. Since the majority of the literatures focus on node classification problem, we give a unified formulation on node-level graph adversarial attacks as an example, which can be easily extended to other tasks.

Definition 4. Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with adjacency matrix $\boldsymbol{A}$ and attribute matrix $\boldsymbol{X}$, let $\mathcal{V}_T$ be the set of nodes to be attacked, the goal of adversarial attack is to find a perturbed graph $\hat{\mathcal{G}}$ that meets the unnoticeable requirement by minimizing the following objective function:

$$\min_{\hat{\mathcal{G}} \in \Phi(\mathcal{G})} \mathcal{L}_{atk}(\hat{\mathcal{G}}) = \sum_{v \in \mathcal{V}_T} l_{atk}(f_{\theta^*}(\hat{\mathcal{G}})_v, y_v)$$
$$\text{s.t.} \quad \theta^* = \arg\min_{\theta} \mathcal{L}_{train}(f_{\theta}(\mathcal{G}')) \qquad (14)$$

where l_{atk} is the loss for attacking, which is typically set as $-H(f_{\theta^*}(\hat{\mathcal{G}})_v, y_v)$ with $H(\cdot)$ being the cross entropy. $\mathcal{L}_{train}$ is the loss for training target model. Generally, in node classification, we apply the classification loss $\mathcal{L}_{train} = \sum_{v \in \mathcal{V}_L} H(f_{\theta}(\hat{\mathcal{G}})_v, y_v)$. As for $\mathcal{G}'$, it can be either $\hat{\mathcal{G}}$ or $\mathcal{G}$, which correspond to poisoning attack and evasion attack, respectively. As for the search space $\Phi(\mathcal{G})$, apart from the way of perturbing the graph, it is also constrained by the budget to ensure unnoticeable attacks. Specially, the constraint from the budget is typically implemented as $\|\hat{\boldsymbol{A}} - \boldsymbol{A}\| + \|\hat{\boldsymbol{X}} - \boldsymbol{X}\| \le \Delta$, where Δ denotes the budget value. For non-targeted attack, $\mathcal{V}_L$ is set all the unlabeled nodes $\mathcal{V}_U$ and y_v will be the prediction of the unlabeled nodes fro a GNN trained on the clean graph $\mathcal{G}$

4.2.2 Evasion attacks

Evasion attacks focus on the inductive setting, which aims to change the predictions on new nodes/graphs. Based on the applied techniques, evasion attack methods can be split into gradient-based methods and reinforcement learning-based methods.

Gradient-based methods. In the early stage of adversarial attacks[30, 115–117], they generally focus on white-box evasion attack to demonstrate the vulnerability of GNNs and assess the robustness of model under worst situations. Since the model parameters are available in white-box evasion attacks, it is natural to optimize the objective function in (14) by gradient decent. More specifically, the objective function of white-box evasion attack can be rewritten as

$$\min_{\hat{\mathcal{G}} \in \Phi(\mathcal{G})} \mathcal{L}_{atk}(\hat{\mathcal{G}}) = \sum_{v \in \mathcal{V}_T} l_{atk}(f_{\theta}(\hat{\mathcal{G}})_v, y_v) \qquad (15)$$

where θ represents mode parameters of the target GNN. However, due to the discreteness of the graph structure, it is challenging to directly solve the optimization problem in (15). Therefore, FGA[116] and GradMax[117] use a gradient-based greedy algorithm to iteratively modify the connectivity of the node pair within attack budget. Xu et al.[115] adopt projected gradient decent to ensure the discreteness of perturbed adjacency matrix. Instead of directly using derivatives from the attacks, Wu et al.[30] use integrated gradients to identify the optimal edges and node attributes to be modified for attack.

Recently, there are several attempts in developing gradient-based evasion attacks[125, 126] in a more practical black-box setting, which can be applied to graph classification task and node classification on evolving graphs. By exploiting the connection between the backward propagation of GNNs and random walks, Ma et al.[126] investigate the connections between the change of classification loss under perturbations and the random walk transition matrix. Based on that, they generalize the white-box gradients into a model-independent important scores of PageRank, which avoids using model parameters. Tao et al.[125] investigate the black-box evasion attack with single node injection. Without knowing the model parameters, they train a surrogate model on the train graph and attack the surrogate model to inject a fake node to the graph. A parameterized attribute generator and edge generator is adopted for the node injection attack on unseen test nodes.

Reinforcement learning-based methods. In black-box evasion attacks, lacking of model parameters will challenge the gradient-based methods. However, in many scenarios such as drug property prediction, the attacker is allowed to query the target model. In this situ-

ation, reinforcement learning can be employed to conduct evasion attacks to learn the optimal actions for graph perturbation[117, 119]. RL-S2V[117] is the first work to apply reinforcement learning for black-box targeted attack. They model the attack process as a Markov decision process (MDP) defined as:

● **State**: The state s_t at time t is represented by the tuple $(\hat{\mathcal{G}}_t, v)$, where $\hat{\mathcal{G}}_t$ is the intermediate modified graph at time t and v is the target node to be attacked.

● **Action**: The attacker of RL-S2V needs to add or delete an edge in each step, which is equivalent to select a node pair. It decomposes the node pair selection action $a_t \in \mathcal{V} \times \mathcal{V}$ into a hierarchical structure of sequentially selecting two nodes, i.e., $a_t = (a_t^{(1)}, a_t^{(2)})$, where $a_t^{(1)}$, $a_t^{(2)} \in \mathcal{V}$.

● **Reward**: The goal of the attack is to fool the target classifier f on the target node v. In RL-S2V, no reward is given in intermediate steps. The non-zero reward is only given at the end as

$$r(s_m, a_m) = \begin{cases} 1, & \text{if } f(\hat{\mathcal{G}}_m)_v \neq y_v \\ -1, & \text{if } f(\hat{\mathcal{G}}_m)_v = y_v. \end{cases} \qquad (16)$$

● **Termination**: The process will stop once the agent modifies m edges.

RL-S2V adopts Q-learning algorithm to solve the MDP problem for targeted attack. A parameterized Q-learning is implemented for better transferablility. Similar reinforcement learning framework is also applied in ReWatt[119]. To make the perturbations more unnoticeable, ReWatt perturbs the graphs by rewiring, i.e., break an edge e_{ij} and rewire the edge to another node to form e_{ik}. Hence, ReWatt employs a different action space that consists of all the valid rewiring operations.

4.2.3 Poisoning attacks by graph manipulation

Apart from the evasion attacks, extensive poisoning attack methods[17, 117, 119] have been investigated for the transductive learning setting, which are more practical for semi-supervised node classification. The majority of them focus on perturbing the graph data by manipulating the original graph[120, 121]. From a technical standpoint, most poisoning attacks through manipulation can be categorized under gradient-based methods.

As (14) shows, the poisoning attack can be formulated as a bilevel optimization problem. To address this problem, various methods[115, 116, 120, 121] that perform gradient-based attacks on a static/dynamic surrogate model have been investigated. For instance, Nettack[120] deploys a tractable surrogate model, i.e., $f_S(\hat{A}, X) = \text{softmax}(\hat{A}^2 X W)$, to conduct poisoning targeted attack. The surrogate model is firstly trained to capture the major information of graph convolutions. Then, the poisoning attack is reformulated to learning perturbations by attacking the surrogate model as

$$\arg\max_{(\hat{A}, \hat{X}) \in \Phi(A, X)} \mathcal{L}(\hat{A}, \hat{X}; W, v) =$$
$$\max_{c \neq y_v} [\hat{A}^2 \hat{X} W]_{vc} - [\hat{A}^2 \hat{X} W]_{vy_v} \qquad (17)$$

where $\Phi(A, X)$ denotes the search space under the unnoticeable constraint, which considers both attack budgets and maintaining important graph properties. To solve (17), Nettack proposes an effective way of evaluating the change of surrogate loss after adding/removing a feature or an edge. The final poisoned graph is obtained by repeatedly selecting the most malicious operation in a greedy search manner until reaching the budget. Similarly, FGA[116] adopts a static surrogate model for poisoning attack. Different from Nettack, FGA directly adopts the GCN model and select the perturbations based on gradients.

As the static surrogate model is trained on the raw graph, which cannot accurately reflect the performance of GNN on the poisoned graph, there are also many works[115, 121] that adopt a dynamic surrogate model to consider the effects of added perturbations to the target model. In [115], min-max topology attack generation is employed for untargeted attack. Specifically, the inner maximization updates the surrogate model on the partial modified graph, and the outer minimization conducts projected gradient decent topology attack on GNN model. Metattack[121] introduces meta-learning to solve the bilevel optimization on the untargeted attack. Essentially, the graph structure matrix is treated as a hyperparameter and the meta-gradient is computed as

$$\nabla_{\mathcal{G}}^{meta} = \nabla_{\mathcal{G}} \mathcal{L}_{atk}(f_{\theta^*}(\mathcal{G})) =$$
$$\nabla_f \mathcal{L}_{atk}(f_{\theta^*}(\mathcal{G})) \cdot [\nabla_{\mathcal{G}} f_{\theta^*}(\mathcal{G}) +$$
$$\nabla_{\theta^*}(f_{\theta^*}(\mathcal{G}) \cdot \nabla_{\mathcal{G}} \theta^*] \qquad (18)$$

where θ^*, i.e., the parameters of surrogate GNN model, is usually obtained by gradient descent in T iterations. For each inner iteration, the gradients of θ_{t+1} is obtained by $\nabla_{\mathcal{G}} \theta_{t+1} = \nabla_{\mathcal{G}} \theta_t - \alpha \nabla_{\mathcal{G}} \nabla_{\theta_t} \mathcal{L}_{train}(f_{\theta_t}(\mathcal{G}))$, where α is the learning rate of the gradient descent in the inner iteration.

Though surrogate model provides a way for poisoning attack, if the target model architecture differs a lot from the surrogate model, the poisoned graph might not able to significantly reduce the performance of the target model when trained on it. Thus, instead of using surrogate models, some works[123, 124] design generalized attack loss functions to poison the graph to improve the transferability. For example, Bojchevski and Günnemann[123] exploit the eigenvalue perturbation theory to efficiently approximate the unsupervised DeepWalk loss to manipulate the graph structure. It further demonstrates that the perturbed graph structure is transferable to various GNN models such as GCN. Graph embedding learning is formulated as a general signal process with corresponding

graph filter in GF-Attack[124], which enables a general attacker that theoretically provides transferability of adversarial samples. GF-Attack is able to perturb both adjacency matrix and feature matrix, which leads to better attacking performance than [123]. Adaptive attacks methods[134] are explored and testified its threat to various defense approaches.

4.2.4 Node injection attacks

Node injection attacks aim to achieve the adversarial goal by adding malicious nodes. This attack process will not affect the existing link structures and node attributes. Hence, node injection is more practical to execute compared with manipulation attacks. A uniform objective function of node injection attack can be formulated as

$$\min_{\mathcal{E}_A, \mathcal{V}_A} \mathcal{L}_{atk}(\hat{\mathcal{G}}) = \sum_{v \in \mathcal{V}_T} l_{atk}(f_{\theta^*}(\hat{\mathcal{G}})_v, y_v)$$
$$\text{s.t.} \quad \theta^* = \arg\min_\theta \mathcal{L}_{train}(f_\theta(\hat{\mathcal{G}})) \qquad (19)$$

where $\mathcal{E}_A$ is the set of edges that link nodes inside $\mathcal{V}_A$ and connect $\mathcal{V}_A$ with clean graph $\mathcal{G}$.

Node injection attack is firstly proposed by NIPA[17], which assigns fake nodes to degrade the overall performance of the target GNN. To achieve stronger attack performance, the attacker in NIPA will also add the labels of the malicious nodes into the training set. The objective function is solved by reinforcement learning, which is similar to RL-S2V that described in Section 4.2.2. But the action space and design of reward are designed for node injection positioning attack. Specifically, in each step, the action is to connect a fake node to the graph and assign a class label to the fake of node to effectively fool GNN trained on the poisoned graph. The action is decomposed into three steps: 1) select a fake node; 2) select a real node and connect to the fake node selected; and 3) assign the label to the selected fake node. The reward is defined based on the performance of the surrogate model trained on the poisoned graph. Let A_t denote the attack success rate on the surrogate model that trained on $\hat{\mathcal{G}}_t$. The reward is defined as $r_t(s_t, a_t) = 1$ if $A_{t+1} > A_t$, where s_t and a_t denote the state and action at time t; Otherwise, $r_t(s_t, a_t) = 0$.

Beyond the practicality, scalability is another advantage of node injection attacks. When we inject a malicious node for adversarial attacks, we only need to consider $d \cdot N$ options, where d and N are the expected degree of the fake node and the graph size. Since d is generally very small and even can be set as one[125], the node injection attack is naturally more scalable than the manipulation attack. Several following works[122, 125, 136] further investigate the node injection attacks for better scalability. AFGSM[122] approximately linearizes the target GCN and derives a closed-form solution for a node injection targeted attack, which has much lower time cost. Experiments on Reddit with over 100K nodes demonstrate its effectiveness and efficiency for targeted attack. G-NIA[125]

explore to conduct targeted attack by injecting a singe node, which avoids the cost of generating multiple nodes for the attack. To efficiently choose nodes connecting with the injected ones, TDGIA[136] introduces a topological defective edge selection strategy. Specifically, a metric based on basic graph structural information is employed to assess nodes most impacted by perturbations in their neighbors. To generate the features for the injected nodes, a the smooth feature optimization objective is designed in TDGIA.

Recently, the limitations of node injections attacks in unnoticeability are explored in [137]. In this work, the authors observed that these injected nodes and edges can disrupt the homophily distribution in the original graphs, compromising its unnoticeability. To solve this issue, Chen et al.[137] propose homophily unnoticeability to regularize the node generation and attachment. In their experiments, the homophily regularization is applied on various existing node injection attacks. Extensive results on many massive size graphs indicate the effectiveness of the proposed homophily regularization.
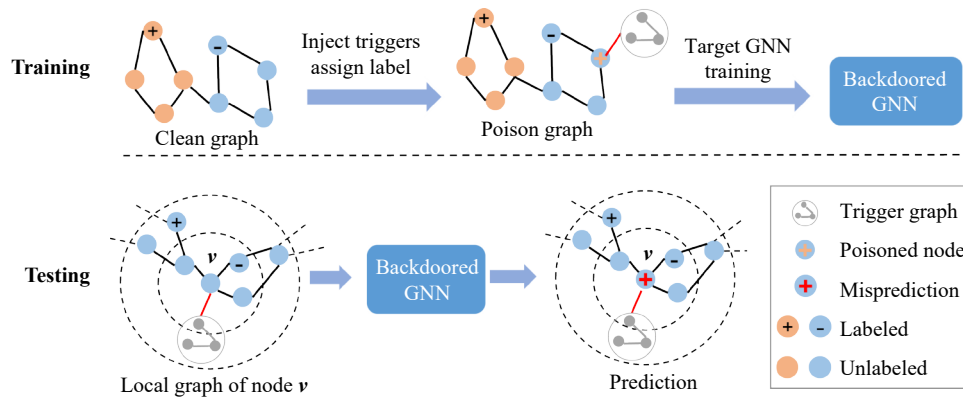
4.2.5 Graph backdoor attacks

In this subsection, we review the recent emerging graph backdoor attacks. In backdoor attacks, the attacker aims to make the host system to misbehave when the pre-defined trigger is present. The backdoor attack on graphs can be applied in various scenarios, which can largely threat the applications of GNNs. For example, an attacker may inject backdoor trigger to the drug evaluation model that possessed by the community. Then, even a useless drug containing the backdoor trigger from the attacker will be classed as an effective medicine. In addition, the backdoor can be applied to federated learning[138], which threatens the safety of the final global model. Though successful backdoor attacks may lead to enormous loss, there are only a few works on backdoor attack methods on graphs[127, 128]. Next, we will introduce a general framework of graph backdoor attacks followed by details of representative works.

General framework of graph backdoor attacks. The key idea of the backdoor attacks is to associate the trigger with the target class in the training data to mislead target models. As illustrated in Fig. 5, during the poisoning phase, the attacker will attach a trigger g to a set of poisoned nodes $\mathcal{V}_P \subseteq \mathcal{V}$ and associate $\mathcal{V}_P$ with the target class label y_t. This process generates a backdoored dataset. The GNNs trained on the backdoored dataset will be optimized to predict the poisoned nodes $\mathcal{V}_P$ (attached with the trigger g) as target class y_t. This association will force the target GNN to relate the existence of the trigger g in neighbors with the target class. In the test phase, the attacker can control the prediction on the target node v to be y_t by attaching the trigger g to node v.

SBA[128] proposes a subgraph-based backdoor attack for graph classification. In [128], the subgraph trigger is

Fig. 5   General framework of graph backdoor attack[135]

generated by random graph generation algorithms such as Erdős-Rényi model and small world model. The set of poisoned nodes $\mathcal{V}_P$ is randomly sampled. In this paper, the authors also investigate the impacts of the trigger size, trigger density, and the size of poisoned graphs, which demonstrates the vulnerability of GNN models to backdoor attacks.

GTA[127] focuses on injecting backdoor to a pre-trained model for node/graph-level tasks. The backdoor is expected to remain effective even after fine-tuning the pretrained model on downstream tasks. Hence, GTA formulates it as the following bi-level optimization problem:

$$\mathcal{G}_t^* = \arg\min_{\mathcal{G}_t} \mathcal{L}_{atk}(\theta^*(\mathcal{G}_t))$$
$$\text{s.t. } \theta^* = \arg\min_{\theta} \mathcal{L}_{train}(\theta, \mathcal{D}_C \cup \hat{\mathcal{D}}_P) \qquad (20)$$

where $\mathcal{D}_C$ and $\hat{\mathcal{D}}_P$ represent the clean training samples and the samples poisoned by $\mathcal{G}_t$, respectively. In [127], the downstream tasks are assumed unknown. They adopt an unsupervised training and design an attack loss that forces the poisoned graphs/nodes have similar embeddings with a pre-defined sample. Equation (20) is solved by iteratively conducting inner and outer optimization with a first-order approximation. In addition, a parameterized backdoor generator is adopted to obtain personalized backdoor for each test graph/node.

UGBA[135]. While SBA and GTA demonstrate impressive attack performance, Dai et al.[135] observed that the triggers they utilize differ significantly from the attached poisoned nodes. This violates the homophily property of the graph. Hence, trigger attachments are vulnerable, as eliminating edges connecting dissimilar nodes can easily disrupt them. Moreover, existing works require expensive budget to backdoor GNNs trained on large datasets. To address these problems, Dai et al.[135] develop UGBA[135] to execute unnoticeable grpah backdoor attacks with limited budget. To make more efficient use of the budget, UGBA will attach triggers with deliberately chosen poisoned nodes denoted as $\mathcal{V}_P$. To ensure unnoticeability, an adaptive trigger generator is deployed to

obtain trigger g_i for node v_i under the following constraint:

$$\min_{(u,v)\in\mathcal{E}_B^i} sim(u,v) \geq T \qquad (21)$$

where $\mathcal{E}_B^i$ denotes the edge set that contains edges inside trigger g_i and edge attaching trigger g_i to node v_i. sim represents the cosine similarity on node features. T is the threshold of the similarity score which can be tuned based on datasets. The objective function of UGBA can be formulated as

$$\min_{\mathcal{V}_P,\theta_g} \mathcal{L}_{atk}(\mathcal{V}_P,\theta_g) = \sum_{v_i\in\mathcal{V}_U} l(f_{\theta^*}(\hat{\mathcal{G}}_i), y_t)$$
$$\text{s.t. } \theta^* = \arg\min_{\theta} \sum_{v_i\in\mathcal{V}_L} l(f_{\theta}(\mathcal{G}_i), y_i) + \sum_{v_i\in\mathcal{V}_P} l(f_{\theta}(\hat{\mathcal{G}}_i), y_t),$$
$$\forall v_i \in \mathcal{V}_P \cup \mathcal{V}_U, \ g_i \text{ meets } (21)$$
$$\text{and } |g_i| < \Delta_g, \ |\mathcal{V}_P| \leq \Delta_P \qquad (22)$$

where $\hat{\mathcal{G}}_i = a(\mathcal{G}_i, g_i)$ denotes the computation graph of node v_i after attaching the generated trigger g_i. $l(\cdot)$ represents the cross entropy loss and θ_g denotes the parameters of the adaptive trigger generator. UGBA splits the optimization process in (22) into poisoned node selection and adaptive trigger generator learning. During the poisoned node selection phase, representative nodes situated at the clustering center are chosen. As for the training of the adaptive generator, a bi-level optimization with a surrogate GCN model is applied.

4.2.6   Scalable attack methods

In real-world scenarios, the graph to be attacked is often in large scale. For example, the Facebook social network contains billions of users. It is challenging to conduct adversarial attacks on such large-scale network. First, the majority of existing works focus on manipulation attacks that try to find the optimal node pairs to be manipulated, which will require very high computation cost. For example, the time and space complexity of Mtattack is $O(N^2)$ with N being the number of nodes in the graph, as it requires to compute the meta-gradient of

each node pair. Second, the large-scale graph may exhibit different properties. Therefore, the attack methods on small graphs could be ineffective on large graphs. However, there are only a few works investigating the vulnerability of GNNs on large-scale graphs[118]. In this section, we present three promising directions of scalable attacks.

Perturbation sampling: As mentioned, the gradient-based manipulation attack needs to compute the gradients of each pair of node to decide the perturbation on the topology, resulting in unaffordable time and space complexity. In [118], projected randomized block coordinate descent (PR-BCD) is proposed to sample the perturbations and update their corresponding probability scores so as to reduce time complexity. Specifically, the manipulation attack on graph topology is modeled as $\min_{\boldsymbol{P}} \mathcal{L}_{atk}(f_\theta(\boldsymbol{A} \oplus \boldsymbol{P}), \boldsymbol{X})$, where $\sum \boldsymbol{P} \leq \Delta$ and $\boldsymbol{P}_{ij} = 1$ denotes an edge flip. The operation $\oplus$ stands for the operation of element-wise edge flipping. To optimize $\boldsymbol{P}$, it is relaxed to $\boldsymbol{P} \in [0,1]^{N \times N}$, where $\boldsymbol{P}_{ij}$ is the probability of flipping the edge for attacks. In each training iteration, PR-BCD randomly samples a fixed number of perturbations, which lead to a sparse adjacency matrix for both forward and backward computation. The final perturbations can be obtained by the flipping probability score matrix $\boldsymbol{P}$. This method is applicable for both targeted and untargeted attacks. Experiments on massive graphs with over 100 million nodes empirically show the effectiveness of PR-BCD and the vulnerability of GNNs on large-scale graphs.

Candidates reduction: For attack on a target node v, the search space can actually be reduced as many candidate topology manipulations are ineffective to affect the prediction on v. For instance, linking two nodes far from the target node v can hardly change the representation of node v. And linking a node u that share the same label as v can even lead to a more robust graph structure[32]. Therefore, SGA[129] develops a mechanism of perturbation candidate reduction to avoid excessive computation. First, for manipulation on a node pair (v_1, v_2), one of the nodes, say v_1, should be in the computation graph of the target node v, i.e., k-hop subgraph of node v in a k-layer GNN. Second, for the other node v_2, its class label should be the one that is most easily misclassified to the original class of target node v. Finally, SGA assumes that a node v_2 is more likely to be selected if node v_2 can largely affect the prediction of v when the manipulation is directly on node pair (v, v_2). Therefore, several best candidates of v_2 for each manipulation can be selected. With these strategies, the time complexity can be reduced to be linear with the graph size.

4.3 Robust graph neural networks

As GNNs are vulnerable to adversarial attacks, various robust graph neural networks against adversarial at-

tacks have been proposed, which can be generally categorized into three types: adversarial training, graph denoising, and certifiable robustness. Next, we will introduce the representative methods of each category and some defense methods lie in other category.

4.3.1 Adversarial training

Adversarial training is a popular and effective approach to defend against adversarial evasion attacks, which has been widely applied in computer vision[139] to defend against evasion attacks. Generally, adversarial training simultaneously generate adversarial samples that can fool a classifier and force the classifier to give similar predictions for a clean sample and its perturbed version so as to improve the robustness of the classifier. Adversarial training[115, 140–144] is also investigated to defend against graph adversarial attacks, which can be generally formulated as the following min max game:

$$\min_\theta \max_{\Delta_{\boldsymbol{A}} \in \mathcal{P}_{\boldsymbol{A}}, \Delta_{\boldsymbol{x}} \in \mathcal{P}_{\boldsymbol{X}}} \mathcal{L}_{train}(f_\theta(\boldsymbol{A} + \Delta_{\boldsymbol{A}}, \boldsymbol{X} + \Delta_{\boldsymbol{X}})) \quad (23)$$

where $\mathcal{L}_{train}$ is the classification loss on the labeled nodes. $\Delta_{\boldsymbol{A}}$ and $\Delta_{\boldsymbol{X}}$ stand for the perturbations on the topology structure and node attributes, respectively. $\mathcal{P}(\boldsymbol{A})$ and $\mathcal{P}(\boldsymbol{X})$ denote the allowable perturbations within the attack budget. Adversarial training on GNN is firstly explored in [115], where the perturbations on graph topology are generated by a PGD algorithm. Some variants of [115] are investigated in [145, 146]. Considering that node classification is a semi-supervised learning task by nature, virtual graph adversarial training[141, 142] is applied to further encourage the smoothness of model predictions on both labeled and unlabeled nodes. In [141, 142], only node feature perturbations are considered in the virtual graph adversarial training as

$$\min_\theta \max_{\Delta_{\boldsymbol{X}} \in \mathcal{P}(\boldsymbol{X})} \mathcal{L}_{train}(f_\theta(\boldsymbol{A}, \boldsymbol{X} + \Delta_{\boldsymbol{X}})) +$$
$$\alpha \sum_{v_i \in \mathcal{V}} D_{KL}(p(y|\boldsymbol{x}_i; \theta) \| p(y|\boldsymbol{x}_i + \Delta_{\boldsymbol{x}_i}; \theta)) \quad (24)$$

where α is the weight of virtual adversarial training regularizer. $p(y|\boldsymbol{x}_i)$ denotes the prediction of node v_i. $D_{KL}(p(y|\boldsymbol{x}_i; \theta) \| p(y|\boldsymbol{x}_i + \Delta_{\boldsymbol{x}_i}; \theta))$ enforces the predictions on unlabeled nodes to be similar with and without perturbation. Recently, a challenging and practical scenario of inductive setting is investigated for adversarial training in [144]. Moreover, learnable diffusion is leveraged in [144] to improve the robustness under adversarial training.

4.3.2 Certifiable robustness

Though various approaches such as graph adversarial training have been proposed to improve robustness against adversarial samples, new attacks may be developed to invalidate the defense methods, leading to an endless arms race. To address this problem, recent works[147, 148] analyze the certifiable robustness of GNNs

to understand how the worst-case attacks will affect the model. Certifiable robustness aims to provide certificates to nodes that are robust to potential perturbations in considered space. For each node $v \in \mathcal{V}$ with label y, the certificate $m(v; \theta)$ can be obtained by solving the following optimization problem:

$$m(v; \theta) = \min_{\hat{\mathcal{G}} \in \Phi(\mathcal{G})} \max_{i \neq y} f_\theta(\hat{\mathcal{G}})_{vy} - f_\theta(\hat{\mathcal{G}})_{vi} \qquad (25)$$

where $f_\theta(\hat{\mathcal{G}})_{vi}$ denotes the predicted logit of node v in class i and $\Phi(\mathcal{G})$ indicates all allowable perturbed versions of the graph. If $m(v) > 0$, the GNN is certifiably robust w.r.t. node v in considered space $\Phi(\mathcal{G})$. In other word, any adversarial samples in $\Phi(\mathcal{G})$ cannot change the prediction to node v from the target model. The work [147] firstly investigates the certifiable robustness against the perturbations on node features. Some following works[148–151] further analyze the certifiable robustness under topology attacks. For instance, Zügner and Günnemann[150] propose a branch-and-bound algorithm that obtains a tight bound on the global optimum of the certificates for topology attacks. In [151], the certificates of a page-rank and a family of GNNs such as APPNP[152] are efficiently computed by exploiting connections to PageRank and Markov decision processes. A technique of randomized smoothing is applied in [148] to give certifiable guarantees to any GNNs. The randomized smoothing will inject noises to the test samples to mitigate the negative effects of the adversarial perturbations. And the obtained certificates are proven to be tight. Recently, a gray-box certificate[153] is proposed to guarantee the robustness to strong adversaries that control entire nodes in the graph. This work introduces a message-interception smoothing strategy and achieves strong robustness theoretically and empirically.

Apart from methods of computing certificates of a trained GNN, robust training that aims to increase the certifiable robustness is also investigated in [147, 151]. The main idea is to directly maximize the worst-case margin $m(v; \theta)$ during training to encourage the model to learn more robust weights. In particular, a robust hinge loss can be added to the training loss to improve certifiable robustness, which can be formally written as

$$\min_\theta \mathcal{L}_{train}(f_\theta(\mathcal{G})) + \sum_{v \in \mathcal{V}} \max(0, M - m(v; \theta)) \qquad (26)$$

where $\mathcal{L}_{train}$ denotes the classification loss, and $M > 0$ is the hyperparameter for the hinge loss. The worst-case margin, i.e., $m(v; \theta)$ is encouraged to be larger than M with (26).

4.3.3 Graph denoising

The adversarial training and certifiable robustness are effective to train robust GNNs to defend against evasion attacks, i.e., attacks happen in the test stage. However, they cannot deal with a poisoned training graphs which

have been perturbed by adversarial attacks. In the early investigation about poisoning attacks[30], topology attack is found to be more effective and in favor by the positioning attacks; while feature-only perturbations generally fail to change the predictions of the target node due to the high dimension of node attribute. Therefore, a promising direction of defensing positioning attacks is to denoise the graph structure to reduce the negative effects of the injected perturbations. Based on the way of denoising the graph, existing methods can be split into pre-processing, graph structure learning, and attention-based methods.

Pre-processing. Pre-processing approaches first denoise the graph using heuristics about network properties or attack behaviors. Then, the GNN model can be trained on the denoised graph to give correct predictions that are not affected by the poisoning attacker. The work [30] firstly proposes a simple and effective pre-processing defense method based on the following two observations on graph adversarial attacks: 1) perturbing graph structures are more effective than modifying the node attributes; and 2) attackers tend to add adversarial edges by linking dissimilar nodes instead of deleting existing edges. Hence, GCN-Jaccard is proposed in [30] to defend against adversarial attacks by eliminating the edges connecting nodes with low Jaccard similarity of node features. Experimental results show the effectiveness and efficiency of this defense method. Apart from the observations about the properties of linked nodes, the adversarial attack is found to result in high-rank spectrum of adjacency matrix, which corresponds to low singular values[154]. Based on the observation of high-rank attack, low-rank approximation with truncated SVD is used to denoise the graph to resist poisoning attacks. Specifically, they retain a truncated SVD that contains only the top-k singular values of the adjacency matrix. Then, the denoised graph can be reconstructed from the truncated SVD. Their experiments show that only keeping the top 10 singular values of the adjacency matrix is able to defend against Nettack[120].

Graph structure learning. Graph structure learning methods[31, 32, 155] aim to simultaneously learn a denoised graph and a GNN model that can give accurate predictions based on the denoised graph. Inspired by the fact that adversarial attacks will lead to high-rank adjacency matrix, Pro-GNN[31] proposes to learn a clean adjacency matrix $\boldsymbol{S}$ with the constraint that 1) $\boldsymbol{S}$ is low-rank and close to the raw perturbed adjacency matrix $\boldsymbol{A}$ such that the adversarial edges are likely to be removed; 2) $\boldsymbol{S}$ should facilitate node classification; and 3) $\boldsymbol{S}$ should maintain feature smoothness, i.e., link nodes of similar features. The overall objective function of Pro-GNN can be written as

$$\min_{\theta, \boldsymbol{S}} \mathcal{L}_{train}(f_\theta(\boldsymbol{X}, \boldsymbol{S})) + \alpha \|\boldsymbol{A} - \boldsymbol{S}\|_F^2 +$$
$$\beta \|\boldsymbol{S}\|_* + \lambda \mathrm{tr}(\boldsymbol{X}^\mathrm{T} \hat{\boldsymbol{L}} \boldsymbol{X}) \qquad (27)$$

where $\mathcal{L}_{train}(f_\theta(\boldsymbol{X}, \boldsymbol{S}))$ is the classification loss using $\boldsymbol{S}$. $\|\boldsymbol{S}\|_*$ stands for the nuclear norm of the learned adjacency matrix $\boldsymbol{S}$ to encourage low-rank of $\boldsymbol{S}$. The last term $\mathrm{tr}(\boldsymbol{X}^{\mathrm{T}} \hat{\boldsymbol{L}} \boldsymbol{X})$ is to encourage the learned adjacency matrix to link nodes of similar features, where $\hat{\boldsymbol{L}}$ is the Laplacian matrix of $\boldsymbol{S}$. Similar to Pro-GNN, PTDNet[155] also adopts a low-rank constraint to learn to drop noisy edges in an end-to-end manner. But different from Pro-GNN which directly optimizes the adjacency matrix of denoised graph, PTDNet deploys a parameterized denoising network to predict whether to remove the edge with the representations of two nodes from a GNN model.

Though the defense methods using low-rank constraint are proven to be effective, the computation cost of nuclear norm is too expensive for large-scale graphs. Recently, a robust structural noise-resistant GNN (RS-GNN)[32] is proposed to learn an link predictor that efficiently eliminate/down-weight the noisy edges with the weak supervision from the adjacency matrix. In real-world graphs, nodes with similar features and labels tend to be linked; while noisy edges would link nodes of dissimilar features. Therefore, RS-GNN deploys an MLP to predict the weight of the link between v_i and v_j by $w(i, j) = f(\boldsymbol{h}_i^{\mathrm{T}} \boldsymbol{h}_j)$, where $\boldsymbol{h}_i = MLP(\boldsymbol{x}_i)$ and f is the activation function such as sigmoid. A novel feature similarity weighted edge-reconstruction loss to train the link predictor to encourage lower weights are assigned to noisy edges. The link prediction is further utilized to predict the missing links in the graph, which can involve more unlabeled nodes in the training to address the challenge of label sparsity.

Reinforcement learning is also applied in graph structure learning for robust representation learning[156]. Specifically, graph denoising policy network (GDPNet)[156] focuses on denoising the one-hop subgraph of each node. Whether to involve the neighbors of a node is determined sequentially. Therefore, the action space would be whether the selected node $u_t \in \mathcal{N}(v)$ at step t should be linked with v. The state $s_t = [\boldsymbol{h}_v^t, \boldsymbol{h}_{u_t}]$ contains the representations of node v by aggregating the previously selected neighbors $\hat{\mathcal{N}}_t(v)$ and the selected node u_t. The prediction scores on the downstream tasks are used as the reward signal for the neighbor selection phase.

Attention mechanism. Attention-based defense methods[157, 158] aim to penalize the weights of adversarial edges or nodes in the aggregation of each GNN layer to learn robust representations. In [157], PA-GNN utilizes auxiliary clean graphs that share the same data distribution with the target poisoned graph to learn to penalize the adversarial edges. Specifically, adversarial edges are injected to the clean graphs to provide supervision for the penalized aggregation mechanism. Let $\mathrm{a}_{ij}^{(l)}$ be the attention score assigned to the edge linking v_i and v_j in the l-th GNN layer. PA-GNN wants the attention weights of clean edges to be larger than the perturbed edges by a margin η as

$$\mathcal{L}_{dist} = -\min\left(\eta, \mathop{\mathbb{E}}_{(v_i.v_j)\in\mathcal{E}_C, 1\le l\le L} \mathrm{a}_{ij}^{(l)} - \mathop{\mathbb{E}}_{(v_i.v_j)\in\mathcal{E}_P, 1\le l\le L} \mathrm{a}_{ij}^{(l)}\right) \tag{28}$$

where $\mathcal{E}_C$ and $\mathcal{E}_P$ are the set of clean edges and perturbed adversarial edges from auxiliary graphs. PA-GNN further adopts Meta-learning to transfer the ability of penalizing adversarial edges to GNN on the target graph. GNNGuard[158] computes the attention scores based on the cosine similarity of node representations from last layer. With the similarity-based attention, the adversarial edges are likely to be assigned with small weights since they generally link dissimilar nodes.

4.3.4 Other defense methods against graph adversarial attacks

In this subsection, we introduce defense methods that do not belong to the above categories.

Robust aggregation. Some efforts[118, 159, 160] have been taken to design a robust aggregation mechanism to restrict the negative effects of perturbations in the graphs. For instance, RGCN[159] adopts Gaussian distributions as the hidden representations of nodes in each graph convolutational layer. As a result, the proposed RGCN could absorb the effects of adversarial changes in the variances of the Gaussian distributions. In [160], a median aggregation mechanism is designed to improve the robustness of GNNs. In median aggregation, the median value of each dimension of the neighbor embeddings is used to capture the context information. Only when the portion of clean nodes in the neighbors is less than 0.5, the perturbed values will be selected in median aggregation, which implies its benefits to the model robustness. Following [160], a soft median aggregation mechanism is applied for scalable defense[118], which computes the weighted mean where the weight of an entry is based on the distance to the dimension-wise median. Extensive experiments on large-scale graphs with up to $100\,\mathrm{M}$ nodes demonstrate the efficiency.

Self-supervised learning defense methods. To address the problem of lacking labels, various self-supervised learning tasks such as link prediction[161] and contrastive learning[162] have been proposed to help representation learning of GNNs. In addition to the prediction accuracy, it is found that some self-supervised tasks can improve the robustness of the GNNs. For instance, SimP-GCN[163] employs a self-supervised similarity preserving task which enforces similar representations for nodes with similar attributes. Therefore, the nodes whose local graph structures are perturbed can still preserve useful representations. In contrastive learning, maximizing the representation consistency between the original graphs and the augmented views of edge perturbation[36, 162] can also result in a more robust model. Some adversarial graph contrastive learning and variants[143, 164–166] are developed to further improve the robustness by introducing an adversarial view of graphs.

4.4 Applications of robust GNNs

Since robust GNNs can defend against adversarial attacks, applications in safety-critical domains will particularly benefit from robust GNNs. For instance, the investigations in bioinformatics graphs such as protein-protein network[44] and brain network[1] require robust GNNs to defend the attacks in bioinformatics[167] to guarantee the safety. In addition, recent research also shows the vulnerability of GNNs in knowledge graph modeling[168]. Nowadays, GNNs have been widely applied to learn useful representations from the knowledge graph to facilitate various downstream tasks such as recommendation system[169]. Therefore, robust GNNs is required for the application of knowledge graphs. Some works have attempted to apply GNNs in financial analysis such as credit estimation and fraud detection[2]. Therefore, robust GNNs are urgent for the security of GNNs in real-world financial analysis.

4.5 Future research directions of robust GNNs

Scalable robust GNNs. As discussed in Section 4.2.6, some initial efforts have verified that adversarial attacks can be applied on extremely large graphs to achieve the attackers′ goal. However, scalable defense methods are rather limited[118]. Though some methods such as GCN-Jaccard is efficient, the defense performance is generally not good enough. For more advanced methods such as Pro-GNN, the computation cost is unaffordable for large-scale graphs. Recently, a reduced-rank topography learning method[170] is proposed to achieve scalable robust graph neural networks. This approach has a nearly-linear complexity in time/space and demonstrate its effectiveness on graphs with millions of nodes. In summary, it is an emerging and promising direction to develop scalable robust GNNs.

Robust GNNs on heterogeneous graphs. Many real-world graphs such as product-user network are heterogeneous, which contain diverse types of objects and relations. Extensive heterogeneous graph neural networks (HGNNs) have been investigated for heterogeneous graphs. However, recent analysis[171] also shows that the adversarial attacks bring more negative effects to metapath-based HGNNs than general GNNs. Despite extensive works on robust GNNs, they are dedicated to homogeneous graphs, which can rarely handle heterogeneous graphs. Thus, developing robust heterogeneous graph neural networks still remains an open problem.

Robust GNNs against label noises. Existing works mainly focus on defending the adversarial attacks on graph structure and node features; while the noises and attacks on labels such as label-flipping attack[172] can also significantly degrade the performance for GNNs. Several initial efforts[36, 172, 173] are conducted to address the challenge of label noises. For instance, Dai et al.[174] firstly develop a label noise-resistant GNN (NRGNN) by linking the unlabeled nodes with (pseudo) labeled nodes with similar features, which can improve the performance of GNNs against label noise or label flipping attack. Though promising, the robust GNNs against label noises is still in an early stage that needs further investigation.

Robust pre-training GNNs. Recently, various pre-training GNN frameworks[132, 162] have been investigated to leverage the large scale of data for downstream tasks. The adversarial attacks can also be applied to the pre-training GNN. For example, backdoor can be injected to a self-supervised learning GNN, which can mislead the model to give target instance even after the fine-tuning[127]. Considering the pre-training GNN model will be utilized to various datasets and downstream tasks, a success adversarial attack on pretraining GNNs could cause huge losses. Thus, it is necessary to develop robust pre-training GNNs.

5 Fairness of graph neural networks

Fairness is one of the most important aspects of trustworthy graph neural network. With the rapid development of graph neural network, GNNs have been adopted to various applications. However, recent evidence shows that similar to machine learning models on i.i.d data, GNNs also can give unfair predictions due to the societal bias in the data. The bias in the training data even can be magnified by the graph topology and message-passing mechanism of GNNs[18]. For example, recommendation system based on random walk is found to prevent females from rising to the most commented and liked profiles[175, 176]. A similar issue has been found in book recommendation with graph neutral network, where the GNN methods could be biased towards suggesting books with male authors[19]. These examples imply that GNNs could be discriminated towards the minority group and hurt the diversity in culture. Moreover, the discrimination would largely limit the wide adoption of GNNs in other domains such as ranking of job applicants[177] and loan fraud detection[43], and even can cause legal issues. Therefore, it is crucial to ensure that GNNs do not exhibit discrimination towards users. Hence, many works have emerged recently to develop fair GNNs to achieve various types of fairness on different tasks. In this section, we will give a comprehensive review of the cutting-edge works on fair GNNs. Specifically, we first introduce the major biases that challenge fairness of GNNs. We then describe various concepts of fairness that are widely adopted in literatures, followed by categorization and introduction of methods for achieving fairness on graph-structured data. Finally, we present public datasets and applications and discuss future directions that need further investigation.

5.1 Biases in graph-structured data and GNNs

Biases widely exist in real-world datasets, which can lead to unfair predictions of machine learning models. Olteanu et al.[178] listed various biases that exist in social data. Suresh and Guttag[179] further discussed various types of biases that cause discrimination issues of machine learning models. According to [177], the bias in machine learning can appear in different stages such as data, algorithm and user interaction. In this paper, we mainly focus on biases in graph-structured data and on GNNs. For a comprehensive review of biases that occur in other phases such as training and evaluation of machine learning models on i.i.d data, please refer to the survey [177].

First, similar to i.i.d data, node attributes/features are often available in graph-structured data. In addition, the data collection of graph-structured data follows similar procedures to i.i.d data such as data sampling and label annotation. Thus, the following biases that widely exist on i.i.d data also exist in graphs.

• **Historical bias**. Various biases such as gender bias and race bias exist in the real world due to historical reasons. These biases can be embedded and reflected in the data. For a system that reflects the world accurately, it can still inflict harm on a population who experience the historical bias[179]. An example of this type of bias is the node embedding learning for link prediction[175]. In particular, users with the same sensitive attributes such as gender and race are more likely to be linked in a real-world graph. As a result, the learned node embeddings will tend to link users with the same gender/race. Then, applications such as friend recommendation that built on these types of node embeddings will reinforce the historical bias.

• **Representation bias.** *Representation bias occurs when the collected samples under-represent some part of the population, and subsequently fails to generalize well for a subset of the use population*[179]. The representation bias can be caused in several ways: 1) the target population does not reflect the use population; 2) the target population under-represents certain groups; and 3) the sampled data does not reflect the target population.

• **Temporal bias**. *Temporal bias arises from differences in populations and behaviors over time*[178]. For graphs, temporal bias can be caused by both the change of node attributes and graph topology. One example is the social network, where the attributes and links of users will evolve over time.

• **Attribute bias**. *Given an attributed network and the corresponding group indicator (w.r.t. the sensitive attribute) for each node, for any attribute, if the distributions between different demographic groups are different, then attribute bias exists in graph*[180]. Attribute bias focuses on the biases in the node attributes of the graphs.

In addition to the aforementioned biases, there are unique types of biases in graph-structured data due to the graph topology:

• **Structural bias**. *We consider an undirected attributed network and the corresponding group indicator (w.r.t. sensitive attribute) for each node. If any information propagation promotes the distribution difference between different groups at any attribute dimension, then structural bias exists in the graph*[180].

• **Linking bias**. *Linking bias arises when network attributes obtained from user connections, activities, or interactions differ and misrepresent the true behavior of the users*[178]. For instance, it is found that younger people are more closely connected than older generations on social network[181]. Moreover, low-degree nodes are more likely to be falsely predicted[182], which leads to degree-related bias.

Generally, GNNs adopt a message-passing mechanism, which aggregates the information of neighbors to enrich the representation of the target nodes. As a result, the learned representations can capture both node attributes and its local topology, which can facilitate various tasks such as node classification. However, due to the biases in topology, the message passing mechanism of GNNs can magnify the biases compared with MLP[18]. In graphs such as social networks, nodes of similar sensitive attributes are more likely to connect to each other[175, 181]. For example, young people tend to build friendship with people of similar age on the social network[181]. The message passing in GNNs will aggregate the neighbor features. Thus, GNNs learn similar representations for nodes of similar sensitive information while different representations for nodes of different sensitive features, leading to severe bias in decision making, i.e., the predictions are highly correlated with the sensitive attributes of the nodes.

5.2 Fairness definitions

In this subsection, we introduce the most widely used fairness definitions, which can be generally split into two categories, i.e., group fairness and individual fairness.

5.2.1 Group fairness

The principle of group fairness is to ensure groups of people with different protected sensitive attributes receive comparable treatments statistically. Various criteria of group fairness have been proposed. Next, we will introduce the mostly used group fairness definitions.

Definition 5 (Statistical parity[183]). Statistical parity, also known as demographic parity, requires the prediction $\hat{y}$ to be independent with the sensitive attribute s, i.e., $\hat{y} \perp s$. The majority of the literatures focus on binary classification and binary attribute, i.e., $y \in \{0, 1\}$ and $s \in \{0, 1\}$. In this situation, statistical parity can be formally written as

$$P(\hat{y} = 1 | s = 0) = P(\hat{y} = 1 | s = 1). \tag{29}$$

According to statistical parity, the membership in the

protected sensitive attributes should have no correlation with the decision from the classifier. Given the definition in (29), the fairness in terms of statistical parity can be measured by

$$\Delta_{SP} = |P(\hat{y} = 1|s = 0) - P(\hat{y} = 1|s = 1)|. \quad (30)$$

A lower Δ_{SP} indicates a more fair classifier. The statistical parity can be easily extended to multi-class and multi-category sensitive attributes problem by ensuring $\hat{y} \perp s$. According to [184], let $y \in \{y_1, \cdots, y_c\}$ and $s \in \{s_1, \cdots, s_k\}$ denotes the multi-class label and multi-category sensitive attribute, where c is the number of classes and k is the number of sensitive attribute categories, the evaluation metric can be extended to:

$$\Delta_{SP} = \frac{1}{k} \sum_{i=1}^{k} \max_{y_j} |P(\hat{y} = y_j) - P(\hat{y} = y_j|s = s_i)|. \quad (31)$$

Statistical parity is the first fairness definition and have been widely adopted. However, some following works[185] argue that statistical parity often cripples the utility of the model. Hence, Equalized Odd is proposed to alleviate the issue, which is defined as:

Definition 6 (Equalized Odds[185]). A predictor satisfies equalized odds with respect to protected attribute s and class label y, if the prediction $\hat{y}$ and s are independent conditioned on y, i.e., $\hat{y} \perp s \| y$. When $y \in \{0, 1\}$ and $s \in \{0, 1\}$, this can be formulated as

$$\forall v \in \{0, 1\}, \ P(\hat{y} = 1|y = v, s = 0) = P(\hat{y} = 1|y = v, s = 1). \quad (32)$$

Equalized odds enforces that the accuracy is equally high in all demographics, punishing models that perform well only on the majority. In the binary classification, we often set $y = 1$ as the "advantaged" outcome, such as "not defaulting on a loan" or "admission to a college". Hence, we can relax the equalized odds to achieve fairness within the "advantaged" outcome group, which is known as equal opportunity.

Definition 7 (Equal Opportunity[185]). It requires that the probability of an instance in a positive class being assigned to a positive outcome should be equal for both subgroup members, i.e.,

$$P(\hat{y} = 1|y = 1, s = 0) = P(\hat{y} = 1|y = 1, s = 1). \quad (33)$$

Equal opportunity expects the classifier to give equal true positive rates across the subgroups, which allows a perfect classifier. Similar to statistical parity, equal opportunity can be measured by

$$\Delta_{EO} = |P(\hat{y} = 1|y = 1, s = 0) - P(\hat{y} = 1|y = 1, s = 1)|. \quad (34)$$

Equalized odds and equal opportunity can be naturally extended to multi-class and multi-category sensitive attributes setting by changing the range of sensitive attributes and labels.

Definition 8 (Dyadic fairness[186]). This can be viewed as an extension of statistical parity for link prediction. It requires the link predictor to give predictions independent with the sensitive attributes of the target nodes. A link prediction algorithm satisfies dyadic fairness if the predictive score satisfies:

$$P(g(u, v)|s(u) = s(v)) = P(g(u, v)|s(u) \neq s(v)) \quad (35)$$

where $g(\cdot)$ is the link predictor, $s(u)$ and $s(v)$ denote the sensitive attributes of node u and v, respectively.

Since the dyadic fairness is extended from the link prediction, the evaluation metric can be simply extended from Δ_{SP} in (31) by replacing the classification probability to link prediction probability, i.e., $\Delta_{DE} = |P(g(u, v)|s(u) = s(v)) - P(g(u, v)|s(u) \neq s(v))|$.

5.2.2 Individual fairness

While group fairness can maintain fair outcomes for a group of people, a model can still behave discriminatorily at the individual level. Individual fairness is based on the understanding that similar individuals should be treated similarly.

Definition 9 (Fairness through awareness[182]). Any two individuals who are similar should receive similar algorithmic outcome. Let $u, v \in \mathcal{X}$ be two data points in dataset $\mathcal{X}$, and $f(\cdot)$ denotes a mapping function. The fairness through awareness can be formulated as

$$D(f(u), f(v)) \leq d(u, v) \quad (36)$$

where $D(\cdot)$ and $d(\cdot)$ are two distance metrics required to be defined in the application context.

Definition 10 (Counterfactual fairness[187]). The counterfactual fairness enforces that predictions for an individual in real-world should remain unchanged in a counterfactual world where the individual's protected attributes had been different. Let $\hat{Y}_{S \leftarrow s}(U)$ and $\hat{Y}_{S \leftarrow s'}(U)$ denote the predictions of a sample with background variable U whose sensitive attributes are set as s and s', respectively. A predictor is counterfactually fair if under any context $X = x$ and $S = s$:

$$P(\hat{Y}_{S \leftarrow s}(U) = y|X = x, S = s) = P(\hat{Y}_{S \leftarrow s'}(U) = y|X = x, S = s) \quad (37)$$

for all y and any value s' of protected attribute S.

The counterfactual fairness can be viewed as individual fairness whose similarity metric treats the individual and its counterfactual sample as a similar pair. How to evaluate individual fairness remains an under-explored research direction. In [188], a measure of individual fair-

ness is proposed. The metrics of group fairness such as Δ_{SP} are also utilized for evaluating the counterfactually fair GNNs[189].

5.3 Fairness-aware graph neural networks

Extensive attempts have been proposed to eliminate the discrimination in machine learning models on i.i.d data[177]. However, these methods cannot be directly applied to graph-structured data because of the unique biases brought by the graph topology and message passing mechanism. Recently, with the remarkable success of GNNs, the concern on fairness issue of GNNs is attracting increasing attention. In this section, we introduce the debiasing methods for achieving fairness in GNNs. Following the categorization of fair machine learning algorithms on i.i.d data[40, 177], existing fairness-aware algorithms can be split into pre-processing methods, in-processing methods, and post-processing methods, based on which stage the debiasing is conducted. Pre-processing approaches are applied to eliminate the bias in data with fair pre-processing methods. In-processing approaches are designed to revise the training of machine learning models to ensure the predictions meet the target fairness definition. Post-processing methods directly change the predictive labels to ensure fairness. Table 7 lists existing works on fair GNNs into these three categories. Based on the techniques they apply, we categorize the debiasing methods on graph-structured data into adversarial debiasing, fairness constraints, and others. Next, we introduce the details of the methods following the categorization based on the techniques.

Table 7 Categorization of fair models on graphs according to the revised stage

Category	References
Pre-processing	[188], [180], [190]
In-processing	[18], [191], [192], [180], [193], [189], [186], [188], [194], [19], [195], [196], [175]
Post-processing	[188]

5.3.1 Adversarial debiasing

Using adversarial learning[197] to eliminate the bias is

firstly investigated in the fair machine learning models on i.i.d data[198–201]. Several efforts[18, 180, 191, 192] have been taken to extend the adversarial debiasing on graph-structured data. An illustration of adversarial debiasing is presented in Fig. 6(a). Generally, adversarial debiasing adopts an adversary f_A to predict sensitive attributes from the representations $\boldsymbol{H}$ of an encoder f_E; while the encoder aims to learn representations that can fool the adversary while can give accurate predictions for the task at hand, say node classification. With the minmax game, the final learned representations will contain no sensitive information, resulting in fair predictions that independent with the sensitive attributes. Thus, statistical parity or dyadic fairness can be guaranteed with the adversarial biasing in node classification and link prediction, respectively. The objective function of adversarial debiasing can be formulated as

$$\min_{\theta_E} \max_{\theta_A} \mathcal{L}_{utility}(f_E(\mathcal{G}; \theta_E)) - \beta \mathcal{L}_{Adv}(f_A(\boldsymbol{H}; \theta_A)) \quad (38)$$

where θ_E and θ_A are parameters of encoder f_E and adversary f_A, respectively. $\mathcal{L}_{utility}$ is the loss function to ensure the utility of the learned representations such as node classification loss and link reconstruction loss. $\mathcal{L}_{Adv}$ is the adversarial loss, which generally is cross entropy loss of sensitive attribute prediction of the adversary based on the learned node representations $\boldsymbol{H}$. β is the hyperparameter to balance the contributions of these two loss terms.

The seminar work [191] firstly applies adversarial debiasing on graph-structured data to learn fair node embeddings. Specifically, it adopts adjacency matrix reconstruction with negative sampling as the utility loss to learn node embeddings. Let $g(v_i, v_j)$ be the predicted probability that v_i and v_j are linked. The utility loss can be written as $\mathcal{L}_{utility} = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}(v_i)} -[\log(g(v_i, v_j)) + \sum_{n=1}^{Q} \mathbb{E}_{v_n \sim P_n(v_i)} \log(1 - g(v_i, v_n))]$, where $\mathcal{N}(v)$ represents the neighbors of node v and $P_n(v_i)$ is the distribution of sampling negative nodes for v_i. Q is the number of negative samples. An MLP is deployed as the adversary f_A to predict the sensitive attributes from the node embeddings $\boldsymbol{H}$. The adversarial loss is given as the binary cross entropy loss between the predictions from

(a) Framework of adversarial debiasing models

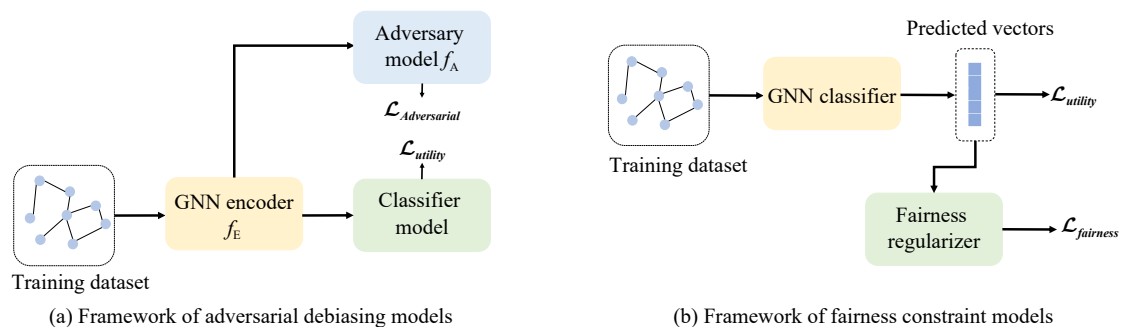(b) Framework of fairness constraint models

Fig. 6 An illustration of fairness-aware GNNs

the adversary and the real sensitive attributes, i.e., $\mathcal{L}_{Adversarial} = \sum_{v \in \mathcal{V}_S} -[s_v \log(\hat{s}_v) + (1 - s_v) \log(1 - \hat{s}_v)]$. Aasrour et al.[192] investigate adversarial debiasing with similar implementation of losses to learn representations for fair link prediction to avoid the separation of users. In addition, they propose a metric to determine whether the predicted links will lead to further separation of the network to evaluate the model.

Though the aforementioned methods achieve fairness with adversarial debiasing, they focus on learning node embeddings and do not use GNN model as the encoder. Recently, FairGNN[18] proposes a framework for fair node classification with graph neural networks. It uses the node classification loss as the utility loss, i.e., $\mathcal{L}_{utility} = \sum_{v \in \mathcal{V}_L} -[y_v \log(\hat{y}_v) + (1 - y_v) \log(1 - \hat{y}_v)]$ where $\mathcal{V}_L$ is the set of labeled nodes. For many real-world applications such as user attribute prediction in social medium, obtaining sensitive attributes of nodes is difficult. To address the challenge of lacking sensitive attributes for adversarial debiasing, FairGNN adopts a GCN-based sensitive attribute estimator to estimate sensitive attributes $\tilde{s}$ for nodes with missing sensitive attributes. It then uses the output of adversary f_A and the estimated sensitive attribute $\tilde{s}$ for adversarial loss. In addition, theoretical analysis in [18] demonstrates that the statistical parity can be guaranteed with adversarial debiasing given the estimated sensitive attributes.

All the aforementioned methods focus on debiasing the node representations. Alternatively, adversarial debiasing can also be applied to debias the original graph data. For example, EDITS[180] utilize a WGAN-based[202] framework to eliminate the attribute bias and structural bias. The attribute matrix $\boldsymbol{X}$ is revised to ensure same

attribute distribution between different demographic groups. Similarly, an adjacency matrix $\boldsymbol{A}$ that will not cause bias after information propagation is learned. More details of the representative adversarial debiasing methods are listed in Table 8.

5.3.2 Fairness constraints

In addition to adversarial debiasing, directly adding fairness constraints to the objective function of machine learning modes is another popular direction. These constraints are usually derived from the fairness definitions introduced in Section 5.2. As the general framework of fairness constrains in Fig. 6(b), they work as the regularization term and balance the performance in prediction and fairness. The overall objective function can be written as

$$\min_{\theta} \mathcal{L}_{utility} + \beta \mathcal{L}_{fairness} \tag{39}$$

where θ is the set of model parameters to be learned, $\mathcal{L}_{utility}$ is the loss function for the utility of the model, $\mathcal{L}_{fairness}$ denotes the applied fairness constraint, and β controls the trade-off between utility and fairness. To enforce different notion of fairness, various constraints have been investigated for fair GNNs[18, 186, 188, 189, 193, 194]. The details of these methods are given in Table 9.

Statistical parity & dyadic fairness. In FairGNN[18], apart from the adversarial debiasing, it also adopts the covariance constraint for statistical parity to further enforce fairness. Specifically, the covariance constraint minimizes the absolute covariance between the estimated sensitive attribute $\tilde{s}$ and predictions $\hat{y}$, i.e., $|Cov(\tilde{s}, \hat{y})| = |\mathbb{E}[(\tilde{s} - \mathbb{E}(\tilde{s}))(\hat{y} - \mathbb{E}(\hat{y}))]|$. Enforcing the predictions to have no correlation with the estimated sensit-

Table 8   Fair graph neural networks that adopt adversarial debiasing

Methods	Task	Fairness	$\mathcal{L}_{Adv}$
FairGNN[18]	Node classification	Statistical parity	Cross entropy between $\hat{s}$ and $\tilde{s}$
EDITS[180]	Node classification	Statistical parity	Wasserstein distance
Compositional[191]	Node classification Link prediction	Statistical parity Dyadic fairness	Cross entropy between $\hat{s}$ and s
FLIP[192]	Link prediction	Dyadic fairness	Cross entropy between $\hat{s}$ and s

Table 9   Fair graph neural networks using fairness constraints

Method	Task	Fairness	$\mathcal{L}_{fairness}$
FairGNN[18]	Node classification	Statistical parity	$|Cov(\hat{y}, \hat{s})|$
UGE[193]	Node classification	Statistical parity	$|g(\boldsymbol{x}_u, \boldsymbol{x}_v) - g(\tilde{\boldsymbol{x}}_u, \tilde{\boldsymbol{x}}_v)|$
FairAdj[186]	Link prediction	Dyadic fairness	$|\mathbb{E}[g(u,v)|s_u = s_v] - \mathbb{E}[g(u,v)|s_u \neq s_v]|$
InFoRM[188]	PageRank spectrual clustering embedding learning	Individual fairness	$\sum_i \sum_j \boldsymbol{S}_{ij}(\boldsymbol{y}_i - \boldsymbol{y}_j)^2$
REDRESS[194]	Node & Link	Individual fairness	Consistency between $\boldsymbol{S}_{\mathcal{G}}$ and $\boldsymbol{S}_{\hat{Y}}$
NIFTY[189]	Node classification	Counterfactual fairness	$|f(\tilde{u}) - f(u)|$

ive attributes will be helpful to learn classifier that give predictions independent with the protected attributes. UGE[193] assumes that a bias-free graph can be generated from the pre-defined non-sensitive attributes. Then, a regularization term pushes the embeddings to satisfy properties of the bias-free graph to eliminate bias. In particular, they enforce $g(\boldsymbol{x}_u, \boldsymbol{x}_v)$, i.e., the probability of predicting links between nodes u and v with complete attributes, is the same as $g(\tilde{\boldsymbol{x}}_u, \tilde{\boldsymbol{x}}_v)$, i.e., the probability of prediction links between nodes u and v with bias-free attributes. In FairAdj[186], a regularization term, $|\mathbb{E}[g(u,v)|s_u = s_v] - \mathbb{E}[g(u,v)|s_u \neq s_v]|$, that directly derived from dynamic fairness based on (35) is used to debias the adjacency matrix.

Individual fairness. Moreover, two constraints for individual fairness are explored in InFoRM[188] and RE-DRESS[194]. InFoRM[188] proposed a regularization term $\sum_i \sum_j \boldsymbol{S}_{ij}(\boldsymbol{y}_i - \boldsymbol{y}_j)^2$, where $\boldsymbol{y}_i \in \mathbf{R}^c$ and $\boldsymbol{y}_j \in \mathbf{R}^c$ denote the prediction vectors of nodes v_i and v_j, and $\boldsymbol{S}_{ij} \in [0,1]$ is the similarity score between v_i and v_j. In this way, for two similar nodes, their predictions will be encouraged to be similar. As for REDRESS[194], it aims to optimize the consistency between the prediction similarity matrix $\boldsymbol{S}_{\hat{Y}}$ and the oracle similarity matrix $\boldsymbol{S}_{\mathcal{G}}$ from a ranking perspective. For each node, the relative order of each node pair by $\boldsymbol{S}_{\hat{Y}}$ and that by $\boldsymbol{S}_{\mathcal{G}}$ are enforced to be the same.

Counterfactual fairness. To achieve counterfactual fairness, NIFTY[189] proposes to maximize the agreement between the original graph and its counterfactual augmented views. More specifically, the counterfactual sample $\tilde{u}$ of an initial sample u is generated by 1) modifying the value of sensitive attribute; and 2) randomly masking the other attributes and perturb the graph structure. Then, the constraint described in Table 9 can be applied to reduce the the gap between the predictions on the original graph and its counterfactual samples, ensuring the counterfactual fairness. Our survey [203] about counterfactual learning gives more details about counterfactual fairness.

5.3.3 Fairness-aware GNNs in other categories

Apart from aforementioned fair GNNs, there are several methods that do not belong to the adversarial debiasing or fairness constraints, which are presented in Table 10. More specifically, users in the same sensitive attribute group are more likely to be sampled into a trace with the generally random walk, resulting to the correlation between the sensitive attributes and predictions. Therefore, unbiased sampling strategies in the random walk are investigated in [175, 196] to learn unbiased embeddings for downstream tasks such as node classification and link prediction. FairDrop[190] proposes to drop more connections between nodes sharing the same sensitive attribute to reduce the bias of homophily in sensitive attributes. Debayes[19] investigates a Bayesian method that is capable of learning debiased embeddings by using

Table 10   Fair GNNs in other categories

Method	Task	Fairness
FairWalk[175]	Link prediction	Statistical parity
CrossWalk[196]	Node & Link	Statistical parity
FairDrop[190]	Node classification	Statistical parity
Debayes[19]	Link prediction	Statistical parity
GMMD[204]	Node classification	Statistical parity

a biased prior. GMMD[204] designs a fairness-awareness message-passing mechanism that will encourage a node to aggregate representations of other nodes from diverse sensitive groups, resulting in fair representations.

5.3.4 Recent advances

In addition to the previously discussed approaches adopting adversarial debiasing and fairness constraints, there are methods emerging in fair augmentation views and enhancement of fairness through model explanations, which are listed in Table 11. Next, we introduce these cutting-edge categories of fair GNNs in details.

Table 11   Recent advances in fair graph neural networks

Category	Task	References
Fair augmentation view methods	Node classification	[189], [205], [206], [207]
	Contrastive learning	[208], [195], [209]
Explanation-enhanced fairness	Node classification	[210], [211]

Fair augmentation view for prediction and contrastive learning. The main idea of the fair augmentation is to generate fairness-aware augmentation views and enforce the agreement between the fair views and original graphs. In this way, the learned representations will achieve fairness and whilst maintaining useful information from original graphs. NIFTY[189] is one of the earliest methods that fail in this category. Specially, counterfactual views of graphs are generated by randomly perturbations on node attributes, sensitive attributes, and graph structures. Then, similarity between the original graph and its counterfactual fair augmented representations is maximized to ensure fairness for node classification. Similar fairness-aware augmented views are further extended to graph contrastive learning for fair node representation learning[195, 208].

Recently, further improvements have been made in the fair augmentation view generation[206, 207, 209, 212]. For example, GEAR[212] considers the sensitive information in the latent space and uses a GraphVAE[205] to model the potential biases from neighboring nodes and the overall graph structure. Through counterfactual data augmentation, GEAR generates perturbed data based on sensitive attributes, aiming to minimize discrepancies between original and counterfactual representations. CAF-GNN[206] uses a matching-based method to find potential views

with different sensitive attributes and labels. Thus, the additional views can be more realistic. Then, CAF-GNN derives proper constraints to ensure the fairness and completeness of the learned representation.

Explanation-enhanced fairness. It has been empirically proven that utilizing graph structure with the messaging-passing mechanism in GNNs can yield bias in predictions[18]. Gaining insights into the origins of such discriminatory predictions is very useful in developing strategies to mitigate biases in GNNs. There are several initial efforts[210, 211] in explaining source of model biases, aiming to enhance the fairness of GNNs through the interpretations to unfairness. Next, we present the existing two works of explanation-enhanced fairness in more details:

• **REFEREE**[210]: While biased GNN predictions can arise from various factors, the biased topology plays a pivotal role in originating and amplifying the discrimination of GNNs. Hence, REFEREE focuses on explaining the unfairness by exploring the edges that maximally account the bias. Let $P(\hat{\mathcal{y}}_0)$ and $P(\hat{\mathcal{y}}_1)$ denote the distributions of predictions on group 0 and group 1, respectively. The problem of finding biased edge set in node v_i's computation graph can be formulated as $\max_{\mathcal{E}_i} W_1(P(\hat{\mathcal{y}}_0), P(\hat{\mathcal{y}}_1))$, where W_1 measures the Wasserstein-1 distance. Similarly, a fair edge set can be found by minimizing the same loss function. Some other regularizations are applied for higher quality of biased and fair edge sets. Fair predictions can be given by removing biased edges while keeping fair edges.

• **BIND**[211]: This approach aims to ascertain the extent to which a GNN's bias is influenced by the presence of a specific training node within the graph. Specifically, BIND uses the influential function to compute the change in the GNN parameters, denoted as $\Delta \boldsymbol{W}$, when node v_i is removed during the training. With the $\Delta \boldsymbol{W}$, the con-

tribution of node v_i to the predictive bias can be quantified. Then, the fairness can be improved by eliminating the nodes that contribute most to the model biases.

5.4 Datasets for fair GNNs

Generally, graph datasets utilized to evaluate the performance of GNNs in terms of fairness need to 1) exhibit bias issue; and 2) have both node label and node sensitive attributes available if the task is node classification. Below, we list some of the widely used datasets that are suitable for evaluating the performance of fair GNNs for node classification and/or link prediction problems. The statistics of the datasets along with papers using these datasets are presented in Table 12.

• **Pokec-n & and Pokec-z**[18]: Pokec-n and Pokec-z datasets collect users' data from Pokec social networks of two provinces in Slovakia in 2012. The Pokec social network is similar to Facebook. Each node in the graph contains attributes such as gender, age, hobbies, interest, education, working field, etc. The datasets target at predicting the occupations of users and the sensitive attribute is region.

• **NBA**[18]: The NBA dataset utilizes 400 NBA players and their social relations on Twitter to construct the graph. The performance statistics of players in the 2016−2017 season and other information, e.g., nationality, age, and salary are provided. The task is to predict whether the players' salary is over median. The sensitive attribute for this dataset is nationality, which is binarized to two categories, i.e., U.S. players and oversea players.

• **German Credit**[189]: The German credit dataset collects data from a German bank[214]. Nodes in the graph represent clients and edges are built between clients if their credit accounts are similar. With clients gender as

Table 12   Datasets for fair graph neural networks

Task	Dataset	Label	Sens.	#Nodes	#Edges	#Features	References
Node classification	Pokec-n	Job	Region	66 569	729 129	59	[18], [193], [195]
	Pokec-z	Job	Region	67 797	882 765	59	[18], [193], [195]
	NBA	Salary	Nationality	403	16 570	39	[18]
	German credit	Credit risk	Gender	1 000	22 242	27	[189], [180]
	Recidivism	Bail	Race	18 876	321 308	18	[189], [180]
	Credit defaulter	Default	Age	3 0000	1 436 858	13	[189], [180]
Link prediction	MovieLens	–	Multi-attribute	9 940	1 000 209	–	[191], [19], [213]
	Reddit	–	Multi-attribute	385 735	7 255 096	–	[191]
	Polblog	–	Community	1 107	19 034	–	[213]
	Twitter	–	Politics	3 560	6 677	–	[196]
	Facebook	–	Gender	22 470	171 002	–	[192], [190], [188], [194], [186]
	Google+	–	Gender	4 938	547 923	–	[192]
	Dutch	–	Gender	26	221	–	[192]

the sensitive attribute, the node classification task aims to predict whether the credit risk of the clients is high or not.

• **Recidivism**[189]: In the Recidivism dataset, nodes are defendants released on bail during 1990–2009[215]. Two nodes are connected if two defendants′ past criminal records and demographics are similar. The task is to predict a defendant as bail (i.e., unlikely to commit a violent crime if released) or no bail (i.e., likely to commit a violent crime) with race being the sensitive attribute.

• **Credit defaulter**[189]: In this dataset, nodes represent credit card users and they are connected based on the similarity of their purchase and payment records. The sensitive attribute of this dataset is age and the task is to classify whether a user will default on credit card payment.

• **MovieLens**[191]: The MovieLens dataset is a recommender system benchmark[216], whose target task is to predict the rating that users assign movies. Sensitive attributes about the user features, such as age, gender, and occupation, are covered in the dataset.

• **Reddit**[191]: The Reddit dataset is based on the social media website Reddit where users can comment on content in different topical communities, called "subreddits". For sensitive attributes, this dataset treats certain subreddit nodes as sensitive nodes, and the sensitive attributes for users are whether they have an edge connecting to these sensitive nodes.

• **Polblog**[213]: Polblog is a blog website network[217]. Nodes represent blogs and links denote hyperlink between blogs. The sensitive attributes for this dataset are blog affiliation communities.

• **Twitter**[196]: This is a subgraph of the Twitter dataset[218, 219]. The sensitive attribute is the political leaning of each user, including neutrals, liberals and conservatives.

• **Facebook**[188]: The dataset is collected from the Facebook website. Nodes and edges represent users and their friendship relations[110]. Gender is used as the sensitive attribute.

• **Google**+[192]: It is collected from Google+[110]. Nodes in the dataset are users who are connected by their social relationships. The sensitive attribute for this dataset is gender.

• **Dutch**[192]: This is from the school network[221] with gender as the sensitive attribute. It corresponds to friendship relations among 26 freshmen at a secondary school.

5.5 Applications of fair GNNs

Social network analysis. With the emerging of social media platforms such as Facebook, Twitter, and Instagram, the social network analysis is widely conducted to provide better service to users. For example, the platforms may use GNNs to recommend new friends to a user[11]. Node classification are also widely conducted on

social networks to further complete user profile for better service[18]. However, recent works[18, 175, 176] indicate that GNNs can be biased to the minority in friends recommendation and node classification on social networks. For instance, the algorithm have been found to prevent minorities from becoming influencers. The message-passing on graphs can magnify the bias[18]. Therefore, several fair GNNs[18, 192, 193, 195, 196] for social network analysis have been proposed.

Recommender system. The user interactions on products such as books can link the users and products to compose bipartite graph. In addition, the social context of users may also be utilized for recommendation. Because the great power of GNN in process graphs, many platforms have applied GNNs for the recommendation system[3, 222]. But fairness issue is also reported in recommendation system. For instance, it is found that a GNN-based algorithm on book recommendation may be biased towards suggesting books with male authors[19]. Hence, it is necessary to develop fair graph neural networks for recommendation system.

Financial analysis. Recently, there is a growing interest in applying GNNs to financial applications such as loan default risk prediction[223, 224] and fraud detection[225]. In loan default risk, the guarantee network[224] or user relational graph[223] can be applied to learn more powerful representations for predictions. In fraud detection, GNNs on transaction[225] are also investigated. Similar to the applications in other domains, GNNs also exhibit bias towards protected attributes such as genders and ages in financial analysis[189]. Using fair GNNs[180, 189] in finance can ensure the fairness to users and avoid the social and legal issues caused by the bias in the GNN model.

5.6 Future research directions of fair graph neural networks

Though many fair models on graph-structured data have been investigated, there are still many important and challenging directions to be explored. Next, we list some promising research directions.

Attack and defense in fairness. Recent works have shown that a poisoning attacker can fool the fair machine learning model to exacerbate the algorithmic bias[21, 22, 226]. For instance, one can generate poisoned data samples by maximizing the covariance between the sensitive attributes and the decision outcome and affect the fairness of the model. Thus, a seriously biased model caused by the attacker might be treated as a fair model by the end-user due to the deployment of fair algorithms, which can result in social, ethical and legal issues. Since GNNs are an extension of deep learning on graphs, fair GNNs are also at a risk of being attacked. Without understanding the vulnerability and robustness of fair GNNs, we cannot fully trust a fair GNN. Despite the initial efforts on attacking fair models[21, 22, 226], all of them

focus on i.i.d data; while the studies on vulnerability of fair GNNs are rather limited. Note that to achieve a trustworthy GNN, the robustness and fairness should be simultaneously meet. However, as it is discussed in Section 4, current robust GNNs generally focus on the robustness in terms of performance and rarely investigate the robust models against attacks in both accuracy and fairness. Therefore, it is crucial to investigate the vulnerability of fair GNNs and develop robust fair GNNs.

Fairness on heterogeneous graphs. Many real-world graphs such as social networks, knowledge graph, and biological networks are heterogeneous graphs, i.e., networks containing diverse types of nodes and/or relationships. Various GNNs have been proposed to address the challenge of representation learning on heterogeneous graphs, such as learning with meta-path[113, 227] and designing new message-passing mechanisms for heterogeneous graphs[228]. Recently, it is reported that the representations learned by heterogeneous GNNs can contain discrimination[229], which could result in societal prejudice in the applications. For example, the social biases have been identified in knowledge graphs[230]. And the representations of knowledge graph learned by heterogeneous GNNs are widely adopted to facilitate the searching and recommender system. Hence, the encoded biases in the representations could lead to detrimental societal consequences. However, existing fair algorithms are generally designed for GNNs on homogeneous graphs, which are not able to mitigate the bias brought by the meth-path neighbors or the message-passing mechanism specifically designed for heterogeneous information networks. Therefore, it is necessary to develop fair GNNs to address the unique challenges brought by the heterogeneity graphs.

Fairness without sensitive attributes. Despite the ability of the aforementioned methods in alleviating the bias issues, they generally require abundant sensitive attributes to achieve fairness; while for many real-world applications, it is difficult to collect sensitive attributes of subjects due to various reasons such as privacy issues, and legal and regulatory restrictions. As a result, most of existing fair GNNs are challenged due to the lacking of sensitive attributes in training data. Though investigating fair models without sensitive attributes is important and challenging, only some initial efforts on i.i.d data have been conducted[231–233]. How to learn fair GNNs without sensitive attributes is a promising research direction.

6 Explainability of graph neural networks

Parallel to the effectiveness and prevalence of deep graph learning systems, the property of lacking interpretability is shared by most deep neural networks (DNNs). DNNs typically stack multiple complex nonlin-

ear layers[234], resulting in predictions difficult to understand. To expose the black box of these highly complex deep models in a systematic and interpretable manner, explainable DNNs[235] have been explored recently. However, most of these works focus on images or texts, which cannot be directly applied to GNNs due to the discreteness of graph topology and the message-passing of GNNs. And it is very important to understand GNNs′ predictions for two reasons. First, it enhances practitioners′ trust in GNN models by enriching their understanding of the network characteristics. Second, it increases the models′ transparency to enable trusted applications in decision-critical fields sensitive to fairness, privacy and safety challenges. High-quality explanations can expose the knowledge captured, helping users to evaluate the existence of possible biases, and make the model more trustworthy. For example, counterfactual explanations are utilized in [236] to analyze the fairness and robustness of black-box models, in order to build a responsible artificial intelligence system. A model-agnostic explanation interface is also designed in [237] to continuously monitor model performance and validate its fairness. Therefore, explainable GNNs are attracting increased attention recently and many efforts have been taken.

In this section, we will provide a comprehensive survey on the current progress in the explainability of GNNs. First, we introduce the background of explaining GNN models and provide a motivating example. Following that, a comprehensive review of existing explanation methods would be presented. Popular datasets and evaluation metrics in this domain are also introduced. Finally, we go through some future research directions. Compared to the existing review in [238], the main improvement of this survey is that we covered more recent progress such as self-explainable GNNs[36, 239] and discussed more reliable evaluation settings[240].

6.1 Backgrounds

6.1.1 Aspects of explanation

As discussed in [241, 242], the term explainability itself needs to be explained. Generally, explainability in GNNs should 1) guide end-users or model designers to understand how the model arrives at its results; 2) enable users to have an expectation on the decisions; 3) and provide information on how and when the trained model might break. To cope with the needs of obtained explanations, explainability must be considered within the context of particular disciplines. As a result, developed explanation methods often show a high level of variety and provide explainability at different levels. Generally, explanations can be categorized from the following perspectives:

● **Global** or **Local** explanations. A local explanation (or instance-level explanation) provides justification for prediction on each specific instance. Currently, most

GNN explanation works[35, 243] fall within this group. On the other hand, global explanations (or model-level explanations)[244] reveal how the model′s inference process works, independently of any particular input.

• **Self-explainable** or **Post-hoc** explanations. Self-explainable GNNs design specific GNN models that are interpretable intrinsically, which can simultaneously give the prediction and corresponding explanation. The explainability arises as part of the prediction process for self-explaining methods. On the contrary, post-hoc explanations focus on providing explanations on a trained model. An additional explainer model is generally adopted for the post-hoc explanations[35]. However, due to the adoption of the explainer model, the post-hoc explanations may misinterpret the actual inner working of the target model.

• **Explanation forms** to be presented to the end users. Explanations should help users to understand GNNs′ behaviors. Various explanation forms have been investigated such as bag-of-edges[245], attributes importance[35], sub-graphs[28], etc. Different explanation forms can give different visualizations and offer different information to end users.

• **Techniques** for deriving the explanations. To enable the explainability, various explanation techniques have been developed including perturbing the input[35], analyzing internal inference process[245], designing intrinsically interpretable GNN models[36], etc. These methods make different assumptions about the model and their advantages may vary across datasets.

A comprehensive taxonomy and detailed introduction of methods are presented in Section 6.3.

6.2 Desired qualities of explanations

With the explanation model and its produced explanations obtained, different aspects regarding explanation quality can be evaluated. Whereas a gold standard exists for comparing predictive models, there is no agreed-upon evaluation strategy for explainable AI methods. As argued in [235], evaluating the plausibility and convincingness of an explanation to humans is different from evaluating its correctness, and those criteria should not be conflated. In this part, we systematically analyze certain properties that good explanations should satisfy. Considerations of these qualities motivate the design of different explanation methods and evaluation metrics, which will be discussed in Sections 6.3 and 6.5, respectively.

• **Correctness**: The obtained explanations should be correct, and truthfully reflect the reasoning of the target predictive model (either locally or globally). This quality addresses the faithfulness of explanations and requires that descriptive accuracy of the explanation is high[246].

• **Completeness**: Completeness addresses the extent to which identified explanations explain the target model. Ideal explanations should contain "the whole truth"[247].

High completeness is desired to provide enough details, and it should be balanced with correctness[235].

• **Consistency**: The obtained explanations should be consistent concerning to inputs. In other words, explaining should be deterministic and the same explanation should be provided for identical inputs[248]. It has also been argued that explanations should be invariant toward small perturbations[249].

• **Contrastivity**: Contrastivity facilitates distinctions of obtained explanations of different predictions. Explanation of a certain event should be discriminative in comparison to those of other events[250]. For models taking different decision strategies, explanations of their behaviors should also be distinct[251].

• **User-friendly**: An ideal explanation is expected to be user-friendly. Explanations should be presented in a form that is clear, easy to interpret, and "agree with human rationales"[252]. For example, it is argued that explanations should be compact, sparse and avoid redundancy[235]. Some formats are also found to be more easily interpretable than others[253].

• **Causality**: A causal explanation provides insights into the cause-and-effect relationships that determines model decisions[254]. Such explanations not only describe the correlations or patterns identified by the model but also delve into the reasons or mechanisms driving those patterns[255]. Ideally, a causal explanation should help users differentiate between spurious correlations and actual causal influences, thereby enabling more reasonable explanations[256].

6.2.1 Explanation example

Due to diverse settings and the complexity of existing algorithms, discussing and comparing GNN explanation methods can often become abstract. To make this explanation task more concrete, we give an example of instance-level explanation, which has been widely taken in various works[28, 35, 243]. As shown in Fig. 7, the explanation objective is to find discriminative substructures, including edges and node attributes, that are important for the prediction of to-be-explained GNN. Note that there are works investigating other explanation forms, like class-wise prototypical structures[239] and interpretable surrogate models[257].

6.2.2 Challenges of explainability of graph neural networks

Besides common difficulties in explaining deep models, there are certain properties making the explanation in graph domain particularly challenging than in other domains like images or texts. First, GNN captures both node attributes and structural topology information, relying upon the message-passing[258] along edges. It is difficult to estimate contribution of edges as they are discrete, and could appear in the computational graphs multiple times at different layers. In turn, identifying important structures is even more complicated as interactions among nodes and edges are involved. Second, graph is
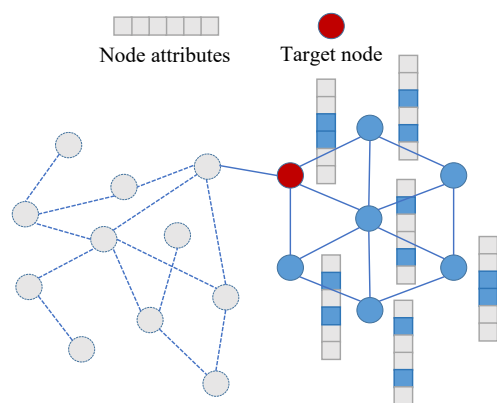
Fig. 7 An example of instance-level explanation, where important nodes, attributes and edges for predictions are highlighted.

less intuitive than images or texts. It is difficult for users to analogize the commonalities and dissimilarities among graphs, making the evaluation and interpretation of obtained explanations (still in the form of graphs) challenging. Domain knowledge often is required to understand the obtained graph "explanations".

6.3 Taxonomy of explainability of graph neural networks

To make deep models applicable in real scenarios, researchers have made extensive attempts to get an explanation from deep models, especially in the image and text domains. However, due to the complexity of graph data and the less human-understandable message-passing mechanism in GNNs, it is difficult to directly extend explanation methods for image or text data to graph data. Recently, to address these challenges, researchers begin to focus on the explainability of GNN models and propose many specific models. In this section, we provide a high-level summary of existing GNN explanation methods and categorize them into three groups: 1) instance-level post-hoc explanations, 2) model-level post-hoc explanations, and 3) self-explainable methods that are intrinsically interpretable. Most existing GNN explanation methods are designed for the instance-level post-hoc setting, and we

further arrange them based on their adopted techniques for achieving explainability. A summary of method taxonomy is shown in Table 13.

6.3.1 Instance-level post-hoc explanation

Instance-level post-hoc explanation identifies elements (like node attributes and edges) that are crucial for model's prediction for each specific input instance. Typically, given an input graph $\mathcal{G}$, which could be a graph sample for graph-level tasks or the local graph of a node for node-level tasks, it aims to find a sub-graph $\mathcal{G}_s \subset \mathcal{G}$ that accounts for prediction output of the target GNN model. Based on different strategies in identifying input substructures as explanations, we can further summarize existing methods into three groups: 1) attribution methods, which directly analyze the influence of input elements on the prediction using gradients or through perturbations; 2) decomposition methods, which examine the inference of deep models by decomposing prediction result into importance mass on the input; 3) surrogate methods, which train an interpretable model that mimics the behavior of to-be-explained deep model within the neighborhood of current input. Next, we introduce the details of these three categories.

Attribution methods. Attribution, also referred to as relevance[263], aims to reveal components of high importance in the input. These methods provide explanations through measuring the contribution of input elements in the target decision, and finding the subgraph with top contribution weights as explanation $\mathcal{G}_s$. Based on their strategies in estimating each element's contribution, we can further categorize them into two types: gradient-based and perturbation-based.

Gradient-based attribution. Gradient-based attribution methods estimate importance weights by back-propagated gradients. Based on Taylor expansion, gradients from model output w.r.t input elements reflect its sensitivity towards them, which can be utilized as an importance estimator[264]. For example, importance of node v with attribute $\boldsymbol{x}_v$ for prediction y^c can be computed as $\|Relu(\frac{\partial y^c}{\boldsymbol{x}_v})\|_1$[259]. One weakness of these methods is that local gradients could be unreliable because of gradient saturation problem[265]. Therefore, integrated gradients (IG)[265] that consider the gradients along a path were

Table 13　Categorization of explanation models on graphs

	Category				References
	Gradient	Perturbation	Decomposition	Surrogate	
Instance-level post-hoc	√				[245], [259]
		√			[28], [35], [243]
			√		[245], [259], [260]
				√	[257], [261], [262]
Model-level post-hoc					[239], [244]
Self-explainable					[36], [239]

proposed to address this problem. On graphs, IG score of node attributes can be computed in the form of:

$$IG(\boldsymbol{x}_v) = (\boldsymbol{x}_v - \boldsymbol{x}'_v) \times \int_{\alpha=0}^{1} \frac{\partial f(\boldsymbol{x}'_v + \alpha \cdot (\boldsymbol{x}_v - \boldsymbol{x}'_v))}{\partial \boldsymbol{x}_v} \mathrm{d}\alpha \tag{40}$$

where $\boldsymbol{x}'_v$ represents the baseline attributes which can be set to the global average. Essentially, (40) integrates the gradients at all points along the path from $\boldsymbol{x}'_v$ to $\boldsymbol{x}_v$, instead of relying on gradient at $\boldsymbol{x}_v$ which may suffer from saturated gradients. Existing works have investigated various gradient-based methods to explain GNNs, such as SA[245], Guided BP[245], CAM[259] and Grad-CAM[259]. These methods share similar ideas to identify important input elements. The main difference lies in the procedure of gradient back-propagation and how different hidden feature maps are combined[238]. For example, Guided BP[245] clips negative gradients during conducting back-propagation to estimate contribution weights. CAM[259] requires that a linear layer is used for classification, and calculates heat maps over nodes using node embeddings from the last GNN layer along with weights of that linear classification layer. Grad-CAM[259] generalizes it to the model-agnostic setting by using average class-wise gradients in place of linear classification parameters. It is straightforward to estimate importance weights of node attributes using these methods, and edges connecting important nodes would also be taken as important[266].

Perturbation-based attribution. Perturbation-based attribution methods try to learn a perturbation mask and examine prediction variations w.r.t perturbations. Specifically, the mask is optimized to maximize the perturbation (mask out as many edges and nodes as possible) while preserving original predictions. Those left-out unperturbed nodes and edges are taken as the most important elements contributing to the prediction, which corresponds to the explanations. Let $\boldsymbol{M}_A \in \{0,1\}^{n \times n}$ and $\boldsymbol{M}_X \in \{0,1\}^{n \times d}$ denote binary masks on edges and node attributes, respectively. It can be formulated as the following optimization problem:

$$\min_{\boldsymbol{M}_A, \boldsymbol{M}_X} \mathcal{L}_{dif}\big(f(\boldsymbol{A}, \boldsymbol{X}), f(\hat{\boldsymbol{A}}, \hat{\boldsymbol{X}})\big) + \beta \cdot \mathcal{R}(\boldsymbol{M}_X, \boldsymbol{M}_A),$$
$$\hat{\boldsymbol{A}} \sim \mathcal{P}(\boldsymbol{A}, \boldsymbol{M}_A), \quad \hat{\boldsymbol{X}} \sim \mathcal{P}(\boldsymbol{X}, \boldsymbol{M}_X). \tag{41}$$

where $\mathcal{P}$ denotes the perturbations on original input with provided importance masks, and we use $\hat{\boldsymbol{A}}$ to represent perturbed $\boldsymbol{A}$. So is the case of $\hat{\boldsymbol{X}}$. With this optimization objective, explanations are found by finding input elements that preserve model predictions. For example, in GNNExplainer[35], $\mathcal{P}(\boldsymbol{A}, \boldsymbol{M}_A) = \boldsymbol{A} \odot \boldsymbol{M}_A$ and $\mathcal{P}(\boldsymbol{X}, \boldsymbol{M}_X) = \boldsymbol{Z} + (\boldsymbol{X} - \boldsymbol{Z}) \odot \boldsymbol{M}_X$, where $\boldsymbol{Z}$ is sampled from marginal distribution of node attributes $\boldsymbol{F}$ and $\odot$ denotes elementwise multiplication. $\mathcal{L}_{dif}$ is usually implemented

as cross entropy loss[35], which encourages consistency on prediction outputs. $\mathcal{R}$ regularizes identified explanations and is usually implemented as sparsity constraint. This objective promotes the correctness and non-redundancy of found explanations.

Equation (41) is a discrete optimization problem, which is difficult to solve directly. Various learning paradigms have been proposed to efficiently find the masks[28, 35, 243]. Based on the adopted strategy in learning perturbation masks, these approaches can be further divided into three groups.

The first group identifies effective perturbation masks by conducting searching[28, 267]. These methods optimize (41) and learn perturbation masks with the explicit test-and-run paradigm. And the optimization directions are found by search algorithms. For example, SubgraphX[28] employs Monte Carlo tree search (MCTS) algorithm to search for the most important subgraph as the explanation of predictions. The Shapley value is used as the measurement of component′s importance during the search phase. ZORRO[267] revises the search process by explicitly encoding fidelity into the objective. Causal screening[268] also falls into this group, which incrementally selects input elements by maximizing individual causal effects at each search step. CF-GNNExplainer[269] focuses on counterfactual explanations, which is derived from identifying minimal perturbations that can change prediction results.

The second group uses attention mechanism to learn perturbation masks[35, 270]. By relaxing binary masks $\boldsymbol{M}_A \in \{0,1\}^{n \times n}$ and $\boldsymbol{M}_X \in \{0,1\}^{n \times d}$ into soft ones, i.e., $\boldsymbol{M}_A \in [0,1]^{n \times n}$ and $\boldsymbol{M}_X \in [0,1]^{n \times d}$, the soft perturbation masks can be directly optimized in an end-to-end manner. For example, GNNExplainer[35] employs a soft mask on attributes by element-wise multiplication and a mask on edges with Gumbel softmax. Then, these two masks are directly optimized with the objective of minimizing size of unperturbed parts while preserving prediction results.

The last group[234, 271] uses an auxiliary model to predict effective perturbation masks with the information from graph and target model. The auxiliary model is optimized with (41) on training samples. And it is assumed to be general and can safely explain new-coming graphs after training. For example, PGExplainer[243] adopts an explanation network g_ϕ to predict preserving probability of each given edge based on node embeddings derived from the target model $f_t(\mathcal{G})$. It can be formally written as $\boldsymbol{M}_A \sim g_\phi(\mathcal{G}, f_t(\mathcal{G}))$. Unlike the previous two strategies, the perturbation mask does not need to be re-learned from scratch for each to-be-explained graph, and this group of methods is much faster to use in the test time. Another representative method is GraphMask[272] which trains a model to produce layer-wise edge perturbation masks. It takes node embeddings from the corresponding layer as input, and provides different edge importance

masks at different propagation steps. Gem[256] introduces the notion of causality into the explanation generation process, which trains a causal explanation model equipped with a loss function based on Granger causality. It has better generalization ability as it has no requirement w.r.t internal GNN structures or knowledge of learning tasks. RCExplainer[254] aims to modeling the common decision logic of GNNs across similar input graphs. This approach ensures noise resistance by leveraging shared decision boundaries, and guarantees counterfactual integrity by ensuring prediction changes upon the removal of identified edges.

Decomposition methods. These methods seek to decompose the prediction of target GNN model into contribution of input features. A contribution score would be assigned to each input element, and an explanation is obtained via identifying inputs with the highest scores. Concretely, the influence mass is back-propagated layer-by-layer onto each input element. And the influence mass from input to output will be decomposed based on neural excitation at each layer. During this process, nonlinear components of the GNN model are generally neglected to ease the problem. Popular decomposition strategies on image data include layer-wise relevance propagation (LRP)[273] and excitation BP[274]. Several efforts have been made to extend them to graphs data[245, 259, 260]. For example, $\alpha\beta$-rule and ϵ-stabilized decomposition rule in original LRP algorithm are extended to work on message passing mechanism of GNN[245]. GNN-LRP[275] evaluates contributions of bag-of-edges and deduces back-propagation rules on graph walks with high-order Taylor decomposition. For example in GCN, as shown in [275], the back-propagation rule decomposing contribution mass of node K at layer $l+1$ to node J at layer l can be written as (42). This algorithm starts by assigning full contribution mass to the target output, and redistributes it with a backward pass on GNN layer by layer. For simplicity, we use $R_K^{l+1,k}$ to represent contribution mass of the k-th dimension of node K's embedding at layer $l+1$, and use $R_{JK}^{l,j}$ to denote its decomposed contribution to the j-th dimension of node J's embedding at layer l. Assuming node embedding dimension to be d, we have:

$$R_{JK}^{l,j} = \sum_{k=1}^{d} \frac{\lambda_{JK} h_J^{l,j} w_{jk}}{\sum_{J \in \mathcal{V}} \sum_{j=1}^{d} \lambda_{JK} h_J^{l,j} w_{jk}} R_K^{l+1,k} \qquad (42)$$

where with $h_J^{l,j}$ is the j-th dimension of node J embedding at layer l, w_{jk} represents the weight linking neuron j in layer l to neuron k in layer $l+1$ which is a scalar, and λ_{JK} is the edge weight connecting node J to node K. Following this rule, contributions can be back-propagated to the inputs, and those nodes with highest R value are preserved as the explanation $\mathcal{G}_s$.

Surrogate methods. Neural networks are treated as black-box models due to their deep architectures and nonlinear operations. It is observed that they hold a highly-complex loss landscape[276] and are challenging to be explained directly. To circumvent nonlinear classification boundary of the trained DNN model, many attempts are made[277] to approximate DNN′s local prediction around each instance $\boldsymbol{x}$ with simple interpretable models such as logistic regression. Specifically, for the target instance $\boldsymbol{x}$, a group of prediction records can be obtained by applying small random noises to it and collecting the target model′s prediction on these perturbed inputs. Then, the interpretable surrogate model can be trained on these records to mimic target model′s behavior locally, which serves as explanations. The searching process of interpretable local model $\xi(\boldsymbol{x})$ can be written as

$$\xi(\boldsymbol{x}) = \arg \min_{f' \in F'} \mathcal{L}(f, f', \pi_{\boldsymbol{x}}) + \Omega(f') \qquad (43)$$

where F' represents candidate interpretable model families and $\pi_{\boldsymbol{x}}$ denotes local neighborhood around instance $\boldsymbol{x}$. $\mathcal{L}(f, f', \pi_{\boldsymbol{x}})$ measures faithfulness of the surrogate model f' in approximating the target model f in the locality $\pi_{\boldsymbol{x}}$. $\Omega()$ measures model complexity to encourage simple surrogate models[277]. Once the surrogate model $\xi(\boldsymbol{x})$ is trained, explanation on the prediction of f on $\boldsymbol{x}$ can be obtained by examining the interpretable function $\xi(\boldsymbol{x})$.

There are several works extending this idea to explain GNNs[257, 261, 262]. They differ from each other mainly in two aspects: the strategy of obtaining local prediction records ($\pi_{\boldsymbol{x}}$), and the interpretable model family (F') selected as candidate surrogates. For example, GraphLime[257] takes neighboring nodes as perturbed inputs. And it employs a nonlinear surrogate model which can assign large weights to features that are important in inference. However, GraphLime ignores graph structures and can only find important node attributes. PGM-Explainer[262] randomly perturbs node attributes to collect local records, and trains a probabilistic graph model (PGM) to fit them. PGM is interpretable and can show the dependency among nodes inside input graph. RelEx[261] randomly samples subgraphs as inputs, and uses GCN as the surrogate model. It can assign importance weights to edges, but requires an additional running of explanation methods on GCN as it is also non-interpretable.

6.3.2 Model-level post-hoc explanation

Compared to instance-level methods, model-level methods focus more on providing general insights by giving high-level explanations that are independent to inputs for deep graph models. The model-level explanations can be representative and discriminative instances for each class[239, 244, 278] or logic rules to depict knowledge captured by deep model[279]. However, due to the highly diverse topology and complex semantics in the graph domain, it is a very challenging task and a few at-

tempts are made to provide model-level post-hoc explanations for GNNs. XGNN[244] aims to expose what input graph patterns can trigger certain predictions of the target GNN model, and adopt a graph generation module to achieve that. They employ input optimization methods and train a graph generator to generate graphs that can maximize the target predictions using reinforcement learning. After training, generated graphs are expected to be representative of each class and can provide global knowledge on the captured knowledge of the target GNN model. Concretely, the desired prototypical explanation for class c is obtained by solving the following objective:

$$\mathcal{G}^* = \arg \max_{\mathcal{G}} P(f_t(\mathcal{G}) = c) \qquad (44)$$

where f_t denotes the target GNN model, and a graph generator is trained for finding $\mathcal{G}^*$ for class c. Another work, GCFExplainer[278], explores the global explainability of GNNs using global counterfactual reasoning, aiming to identify a concise set of representative counterfactual graphs that elucidate all input graphs. To achieve this, they employ vertex-reinforced random walks on a graph′s edit map and use a greedy technique to get the summarization for each class.

6.3.3 Self-explainable approaches

Different from the post-hoc explanation, self-explainable approaches[36, 239, 280] aim to give predictions and provide explanations for each prediction simultaneously. Specific GNN architectures are adopted to support built-in interpretablity, similar to attention mechanism in GAT[270]. However, although these methods are explainable by design, they are often restricted in the modeling space and struggle to generalize across tasks at the same time. There are several representative self-explainable GNNs proposed recently, i.e., SE-GNN[36], GIB[280], and ProtGNN[239]. Many causal-based methods can also be categorized into self-explainable methods, i.e., DIR[281], DisC[282] and CIGA[283].

The first group of algorithms provides instance-level self-explanation[36, 280]. SE-GNN[32] obtains the self-explanation for node classification via identifying interpretable K-nearest labeled nodes for each node and utilizes the K-nearest labeled nodes to simultaneously give label prediction and explain why such prediction is given. More specifically, SE-GNN[36] adopts an interpretable similarity modeling to compute the attribute similarity and local structure similarity between the target nodes and labeled nodes. A contrastive pretext task is further deployed in SE-GNN to provide self-supervision for interpretable similarity metric learning. GIB[280] balances expressiveness and robustness of the learned graph representation by learning the minimal sufficient representation for a given task. Following the general information bottleneck, it maximizes the mutual information between the representation and the target, and simultaneously constrains the mutual information between the represent-

ation and the input data.

Model-level self-explainable model is also explored in recent years. ProtGNN[239] is better at global-level explanations by finding several prototypes for each class. Newly-coming instances are classified via comparing with those prototypes in the embedding space. A conditional subgraph sampling module is designed to conduct subgraph-level matching and several regularization terms are used to promote diversity of prototypical embeddings. DIR[281] utilizes the intrinsic interpretability of graph neural networks and aim to identify a subset of input graph features, termed "rationale", to guide the model predictions. They formulate the problem into a invariant learning problem and design distribution intervener to get the rationales. DisC[282] is a disentangled GNN framework that separates input graphs into causal and bias substructures, using a parameterized edge mask generator and training two GNN modules with respective loss functions. CIGA[283] provides an alternative causal-based framework, which can capture graph invariance for reliable OOD generalization across diverse distribution shifts. CIGA utilizes an information-theoretic objective to identify subgraphs enriched with invariant intra-class information, ensuring resilience to distribution shifts.

6.4 Datasets for explainability of graph neural networks

For comparing various explanation methods, it is desired to have datasets where the rationale between input graphs and output labels are intuitive and easy-to-obtain, so that it would be easier to evaluate identified explanatory substructures. In this subsection, we summarize popular datasets used by existing works on explainable GNNs, which can roughly be categorized into synthetic datasets and real-world datasets. Several representative benchmarks of both groups will be introduced in detail.

6.4.1 Synthetic data

With carefully-designed graph generation mechanism, we can constrain unique causal relations between input elements and provided labels in synthetic datasets. GNN models must capture such patterns for a successful training and obtained explanations are evaluated with those ground-truth causal substructures. Several common synthetic datasets are listed below:

• **BA-Shapes**[35]: It is a single graph consisting of a base Barabasi-Albert (BA) graph (contains 300 nodes) and 80 "house"-structured motifs (contains 5 nodes). "House" motifs are randomly attached to the base BA graph. Nodes in the base graph are labeled as 0 and those in the motifs are labeled as $1, 2,$ and 3 based on their positions. Explanations are evaluated on attached nodes of motifs, with edges inside the corresponding motif as ground-truth.

• **Tree-Cycles**[35]: It is a single network with a 8-layer balanced binary tree as the base graph. 80 cycle mo-

tifs (contains 6 nodes) are randomly attached to the base graph. Nodes in the base graph are labeled as 0 and those in the motifs are labeled as 1. Ground-truth explanations for nodes within cycle motifs are provided for evaluation.

● **BA-2motifs**[35]: This is a graph classification dataset containing 800 graphs. Half of the graphs are constructed by attaching a "house" motif to BA base graphs, while the other half graphs attach a five-node cycle motif. A binary label is assigned to each graph according to its attached motif. The motif serves as the ground-truth explanation.

● **Infection**[240]: This is a single network initialized with a ER random graph. 5% of nodes are labeled as infected (class 0), and the remaining nodes are labeled as their shortest distances to those infected ones. In evaluation, nodes with multiple shortest paths are discarded. All remaining nodes have one distinct path as the oracle explanation towards their labels.

● **Syn-Cora**[36]: This is synthesized from the Cora[4] to provide ground-truth of explanations, i.e., K-nearest labeled nodes and edge machining results. To construct the graph, motifs are obtained by sampling local graphs of nodes from Cora. Various levels of noises are applied to the motifs in attributes and structures to generate similar local graphs.

6.4.2 Real-world data

Due to high complexity in patterns and possible existence of noises, it is challenging to obtain human-understandable rationale from node features and graph topology to labels for real-world graphs. Typically, strong domain knowledge is needed. Thus, real-world graph datasets with ground-truth explanations are limited. Below are two benchmark datasets:

● **Molecule data**[286]: This is a graph classification dataset. Each graph corresponds to a molecule with nodes representing atoms and edges for chemical bonds. Molecules are labeled with consideration of their chemical properties, and discriminative chemical groups are identified using prior domain knowledge. Chemical groups

NH_2 and NO_2 are used as ground-truth explanations.

● **Sentiment graphs**[238]: It contains three graph classification datasets created from text datasets for sentiment analysis, i.e., Graph-SST3, Graph-SST5 and Graph-Twitter. Each graph is a text document where nodes represent words and edges represent relationships between word pairs constructed from parsing trees. Node attributes are set as word embeddings from BERT[287]. There is no ground-truth explanation provided. Heuristic metrics are usually adopted for evaluation.

A summary of representative benchmark datasets is provided in Table 14 along with their key statistics, tasks and papers used the datasets.

6.5 Evaluation metrics

Besides visualizing identified explanations and conducting expert examinations, several metrics have been proposed for quantitatively evaluating explanation approaches from different perspectives. Next, we would present the major categories of metrics used and introduce their distinctions.

● **Explanation accuracy.** For graphs with ground-truth rationale known, one direct evaluation method is to compare identified explanatory parts with the real causes of the label[35, 243]. F1 score and ROC-AUC score can both be computed on identified edges. The higher these scores are, the more accurate the obtained explanation is.

● **Explanation fidelity.** In the absence of ground-truth explanations, heuristic metrics can be designed to measure the fidelity of identified substructures[238]. The basic idea is that explanatory substructures should play a more important role in predictions. Concretely, Fidelity+[238] is computed by removing all input elements first, then gradually adding features with the highest explanation scores. Heuristically, a faster increase in GNN′s prediction indicates stronger fidelity of obtained explanations. In turn, fidelity[238] is computed by sequentially removing edges following assigned importance weight. In

Table 14   Datasets for explainability of GNNs

Tasks	Dataset	Graphs	Avg.Nodes	Avg.Edges	Features	References
Node classification	BA-shapes	1	700	4 110	10	[35], [243], [28], [262], [36], [239]
	BA-community	1	1 400	8 920	1	[35], [243], [267]
	Tree-cycles	1	871	1 950	10	[35], [243], [262]
	Tree-grid	1	1 231	3 410	10	[35], [243], [262]
	Syn-cora	1	1 895	2 769	1 433	[36]
Graph classification	BA-2motifs	1 000	25	51.4	10	[243], [28]
	Infection	10	1 000	3 996	2	[240], [243]
	Graph-SST2	70 042	10.199	9.20	768	[238], [28], [239]
	Graph-SST5	11 855	19.849	18.849	768	[238]
	Graph-Twitter	6 940	21.103	21.10	768	[238], [239]
	MUTAG	188	19.79	17.93	14	[285], [35], [243], [28], [261], [244], [239]

turn, a faster performance drop represents stronger fidelity of removed explanations.

● **Sparsity.** Good explanations are expected to be minimal structure explanations, as only the most important input features should be identified. This criteria directly measures sparsity of obtained explanation weights[288], and a better explanation should be sparser.

● **Explanation stability.** As good explanations should capture the intrinsic rationale between input graphs and their labels, this criterion requires identified explanations to be stable with respect to small perturbations[189]. Stability score can be computed via comparing explanation changes after attaching new nodes or edges to the original graph. However, GNN′s prediction is sensitive towards input. Thus, it is challenging to select a proper amount of perturbations.

6.6 Future research directions on explainability of GNNs

In this part, we provide our opinions on some promising future research directions. We hope it can inspire and encourage the community to work on bridging the gaps in GNN interpretation.

Class-level explanations. Despite many interpretation approaches, class-wise explanations remain an underexplored area. Instance-level explanations provide a local view of GNN′s prediction. However, it is important to recognize that they may provide anecdotal evidence and scale poorly to large graph sets. On the other hand, class explanations could provide users with both a global view of model′s behavior and a discriminative view grounded in each class, making it easier to expose and evaluate learned knowledge.

Benchmark datasets for interpretability. One great obstacle in designing and evaluating interpretation methods is the measurement of provided interpretability. To a large extent, the difficulty is inherent. For example, it is impossible to provide gold labels for what is a correct explanation. After all, we would not need to design an explanation approach in the first place if we have already known those real explanations. Currently, we rely upon heuristic metrics and approximations, but a set of principled proxy measurements and benchmark datasets are yet to be established.

User-oriented explanations. The purpose of designing explanation approaches is to use them on real-world tasks to expose learned knowledge of trained models. Based on the needs of users, different requirements may arise in the form of explanation and levels of interpretability. Hence, we encourage researchers to consider real user cases, select suitable design choices, and evaluate explanation algorithms in real-world scenarios for the integrity in interpretability field. One promising direction is to provide flexible fine-grained multi-level explanations so that end-users can select the level of explanations they can understand or satisfy their criteria.

Causal and counterfactual explanations. While traditional explanations offer insights into model behavior, causal and counterfactual explanations dig deeper into understanding the cause-and-effect relationships that drive predictions. Causal explanations focus on identifying the direct influences or drivers behind a particular prediction, while counterfactual explanations provide insights by answering "what if" scenarios, showcasing how a prediction might change if input features were altered. Incorporating such explanations in GNNs can be beneficial, especially in critical applications where understanding the underlying causes or potential outcomes of a decision is crucial. It is paramount to develop methods that can reliably extract and present these types of explanations, ensuring that users understand not only what the model predicts but also the underlying reasons and potential alternatives.

7 Future research directions of trustworthy graph learning

Unified framework for trustworthy GNNs. High-stake applications, such as financial analysis, often require multiple dimensions of trustworthiness. However, existing research in trustworthy GNNs mainly focuses on a single dimension of trustworthiness. Methods designed for distinct trustworthy aspects can be in conflict with each other in model design, resulting in poor performance. Consequently, a unified framework for trustworthy GNNs is necessary. Here, we discuss how different dimensions of trustworthiness can interact with each other and briefly review existing related works.

Firstly, there is overlap between fairness and privacy. To learn embeddings that are fair towards sensitive attributes, sensitive attribute information will be eliminated from the embeddings[18]. This can simultaneously protect the privacy of the sensitive information. To further reduce the risk of private information leakage from centralized storage, a local differential privacy mechanism is deployed to the FairGNN in [289]. The threat to fairness from adversarial attacks[290, 291] are also promising directions to investigate. As for the privacy and robustness, RM-GIB[93] proposes a graph information bottleneck framework to simultaneously defend the graph adversarial attacks and membership inference attacks. What′s more, the explanations of graph models can benefit the robustness and fairness. For instance, the explanations may provide useful information to detect the adversarial perturbations, noisy labels and anomaly samples[292]. There are also initial efforts[210, 211] in explaining source of model biases, aiming to enhance the fairness of GNNs.

Trustworthy model for science. Graph-structured data is also very pervasive in scientific research such as the studies on molecules, proteins, and gene graphs. Cap-

turing useful information from such complex topology is a crucial challenge. For instance, the 3D topology of a drug molecule plays a vital role in drug analysis. Hence, adopting trustworthy graph models to process these data could bring breakthroughs to scientific problems in critical areas. Some preliminary works have been conducted in explaining the analysis results in scientific problems such as molecule property prediction[32] and protein structure prediction[293]. But there are still many problems to be solved in trustworthy model for science.

Graph foundation models and trustworthiness. Recent breakthroughs in large language models demonstrate the power of large foundation models trained on massive data. Foundation models trained on large-scale image/text data can consistently enhance the performance. Inspired by this success, learning graph foundation models on vast graph data is promising to attain a level of expertise ready for the deployment in production environments. Some investigations[294–296] have been conducted to achieve graph foundation models. Given the wide applicability of foundational models across tasks, ensuring their trustworthiness is paramount. Some recent works have explored the trustworthiness in large-scale pretraining, which can provide some insights for trustworthy graph foundation models. For example, fairness-aware augmented views are employed in graph contrastive learning for fair node representation learning[195, 208]. For the robustness perspective, unsupervised anomaly detection on large power grids[220] and certified robustness on graph contrastive learning[284] are also explored. However, the investigations of graph foundation models and its trustworthiness are still in an early stage, which requires further research and development.

8 Conclusions

In this survey, we conduct a comprehensive review on the trustworthy GNNs from the aspects of privacy, robustness, fairness, and explainability. This fills the gap in lacking systematically summary about privacy-preserving GNNs and fairness-aware GNNs. For the robustness and explainability, we introduce the recent trends in more details apart from the representative methods that are reviewed before. More specifically, for each aspect, we introduce the core definitions and concepts to help the readers to understand the defined problems. The introduced methods are categorized from various perspectives. The unified framework of each category is generally given followed by the detailed implementations of the representative methods. In addition, we also list the used datasets in privacy, fairness, and explainability, where the proposed methods have special requirements on the datasets to be trained or evaluated. Numerical real-world applications of trustworthy GNNs are also provided to encourage the researcher to develop practical trustworthy GNNs. Finally, we discuss the future research directions

of each aspect at the end of each section, which includes promising directions in a single aspect and interactions between aspects for trustworthy GNNs.

Acknowledgements

Declarations of conflict of interest

Jiliang Tang is an Associate Editor for *Machine Intelligence Research* and was not involved in the editorial review, or the decision to publish this article. All authors declare that there are no other competing interests.

Open Access

References

[1] J. Kawahara, C. J. Brown, S. P. Miller, B. G. Booth, V. Chau, R. E. Grunau, J. G. Zwicker, G. Hamarneh. Brainnetcnn: Convolutional neural networks for brain networks; towards predicting neurodevelopment. *NeuroImage*, vol. 146, pp. 1038–1049, 2017. DOI: 10.1016/j.neuroimage.2016.09.046.

[2] J. N. Wang, S. Zhang, Y. H. Xiao, R. Song. A review on graph neural network methods in financial applications. *Journal of Data Science*, vol. 20, no. 2, pp. 111–134, 2022. DOI: 10.6339/22-JDS1047.

[3] W. L. Hamilton, R. Ying, J. Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, USA, pp. 1025–1035, 2017.

[4] T. N. Kipf, M. Welling. Semi-supervised classification

with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France, 2017.

[5] J. Chen, T. F. Ma, C. Xiao. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada, 2018.

[6] T. Xiao, Z. Y. Chen, D. L. Wang, S. H. Wang. Learning how to propagate messages in graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ACM, Singapore, pp. 1894–1903, 2021. DOI: 10.1145/3447548.3467451.

[7] T. X. Zhao, X. Zhang, S. H. Wang. Exploring edge disentanglement for node classification. In *Proceedings of ACM Web Conference 2022*, ACM, Lyon, France, pp. 1028–1036, 2022. DOI: 10.1145/3485447.3511929.

[8] L. Lv, J. B. Cheng, N. B. Peng, M. Fan, D. B. Zhao, J. H. Zhang. Auto-encoder based graph convolutional networks for online financial anti-fraud. In *Proceedings of IEEE Conference on Computational Intelligence for Financial Engineering & Economics*, IEEE, Shenzhen, China, 2019. DOI: 10.1109/CIFEr.2019.8759109.

[9] M. Harl, S. Weinzierl, M. Stierle, M. Matzner. Explainable predictive business process monitoring using gated graph neural networks. *Journal of Decision Systems*, vol. 29, no. S1, pp. 312–327, 2020. DOI: 10.1080/12460125.2020.1780780.

[10] X. X. Li, Y. Zhou, N. Dvornek, M. H. Zhang, S. Y. Gao, J. T. Zhuang, D. Scheinost, L. H. Staib, P. Ventola, J. S. Duncan. BrainGNN: Interpretable brain graph neural network for fMRI analysis. *Medical Image Analysis*, vol. 74, Article number 102233, 2021. DOI: 10.1016/j.media.2021.102233.

[11] W. Q. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. L. Tang, D. W. Yin. Graph neural networks for social recommendation. In *Proceedings of World Wide Web Conference*, ACM, San Francisco, USA, pp. 417–426, 2019. DOI: 10.1145/3308558.3313488.

[12] Q. Y. Tan, N. H. Liu, X. Hu. Deep representation learning for social network analysis. *Frontiers in Big Data*, vol. 2, Article number 2, 2019. DOI: 10.3389/fdata.2019.00002.

[13] J. Y. Chen, Z. Q. Shi, Y. Y. Wu, X. H. Xu, H. B. Zheng. Link prediction adversarial attack, [Online], Available: https://arxiv.org/abs/1810.01110, 2018.

[14] M. Jin, H. Chang, W. W. Zhu, S. Sojoudi. Power up! Robust graph convolutional network against evasion attacks based on graph powering, [Online], Available: https://arxiv.org/abs/1905.10029, 2019.

[15] X. L. He, J. Y. Jia, M. Backes, N. Z. Gong, Y. Zhang. Stealing links from graph neural networks. In *Proceedings of the 30th USENIX Security Symposium*, pp. 2669–2686, 2021.

[16] Z. K. Zhang, M. Chen, M. Backes, Y. Shen, Y. Zhang. Inference attacks against graph neural networks. In *Proceedings of the 31st USENIX Security Symposium*, Boston, USA, pp. 4543–4560, 2022.

[17] Y. W. Sun, S. H. Wang, X. F. Tang, T. Y. Hsieh, V. Honavar. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of Web Conference 2020*, ACM,

Taipei, China, pp. 673–683, 2020. DOI: 10.1145/3366423.3380149.

[18] E. Y. Dai, S. H. Wang. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, ACM, Israel, pp. 680–688, 2021. DOI: 10.1145/3437963.3441752.

[19] M. Buyl, T. De Bie. DeBayes: A Bayesian method for debiasing network embeddings. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 1220–1229, 2020.

[20] N. Smuha. Ethics Guidelines for Trustworthy AI. In *AI & Ethics, Date: 2019/05/28-2019/05/28, Location: Brussels (Digityser)*, Belgium, 2019.

[21] D. Solans, B. Biggio, C. Castillo. Poisoning attacks on algorithmic fairness. In *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, Ghent, Belgium, pp. 162–177, 2020. DOI: 10.1007/978-3-030-67658-2_10.

[22] N. Mehrabi, M. Naveed, F. Morstatter, A. Galstyan. Exacerbating algorithmic bias through fairness attacks. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pp. 8930–8938, 2021. DOI: 10.1609/aaai.v35i10.17080.

[23] W. Q. Fan, W. Jin, X. R. Liu, H. Xu, X. F. Tang, S. H. Wang, Q. Li, J. L. Tang, J. P Wang, C. Aggarwal. Jointly attacking graph neural network and its explanations, [Online], Available: https://arxiv.org/abs/2108.03388, 2021.

[24] Y. Zhang, Y. Y. Zhao, Z. Q. Li, X. Q. Cheng, Y. Wang, O. Kotevska, P. S. Yu, T. Derr. A survey on privacy in graph neural networks: Attacks, preservation, and applications, [Online], Available: https://arxiv.org/abs/2308.16375, 2023.

[25] A. Said, T. Derr, M. Shabbir, W. Abbas, X. Koutsoukos. A survey of graph unlearning, [Online], Available: https://arxiv.org/abs/2310.02164, 2023.

[26] W. Jin, Y. X. Li, H. Xu, Y. Q. Wang, S. W. Ji, C. Aggarwal, J. L. Tang. Adversarial attacks and defenses on graphs: A review, a tool and empirical studies, [Online], Available: https://arxiv.org/abs/2003.00653, 2020.

[27] Y. S. Dong, J. Ma, S. Wang, C. Chen, J. D. Li. Fairness in graph mining: A survey. *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 10, pp. 10583–10602, 2023. DOI: 10.1109/TKDE.2023.3265598.

[28] H. Yuan, H. Y. Yu, J. Wang, K. Li, S. W. Ji. On explainability of graph neural networks via subgraph explorations. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 12241–12252, 2021.

[29] H. Zhang, B. Wu, X. L. Yuan, S. R. Pan, H. H. Tong, J. Pei. Trustworthy graph neural networks: Aspects, methods, and trends. *Proceedings of the IEEE*, vol. 112, no. 2, pp. 97–139, 2024. DOI: 10.1109/JPROC.2024.3369017.

[30] H. J. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, L. M. Zhu. Adversarial examples for graph data: Deep insights into attack and defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, Macao, China, pp. 4816–4823, 2019. DOI: 10.24963/ijcai.2019/669.

[31] W. Jin, Y. Ma, X. R. Liu, X. F. Tang, S. H. Wang, J. L.

Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, USA, pp. 66–74, 2020. DOI: 10.1145/3394486.3403049.

[32] E. Y. Dai, W. Jin, H. Liu, S. H. Wang. Towards robust graph neural networks for noisy graphs with sparse labels. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, ACM, USA, pp. 181–191, 2022. DOI: 10.1145/3488560.3498408.

[33] P. Y. Liao, H. Zhao, K. Y. L. Xu, T. Jaakkola, G. J. Gordon, S. Jegelka, R. Salakhutdinov. Information obfuscation of graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 6600–6610, 2021.

[34] C. H. Wu, F. Z. Wu, Y. Cao, Y. F. Huang, X. Xie. FedGNN: Federated graph neural network for privacy-preserving recommendation, [Online], Available: https://arxiv.org/abs/2102.04925, 2021.

[35] R. Ying, D. Bourgeois, J. X. You, M. Zitnik, J. Leskovec. GNNExplainer: Generating explanations for graph neural networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 829, 2019.

[36] E. Y. Dai, S. H. Wang. Towards self-explainable graph neural network. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ACM, Australia, pp. 302–311, 2021. DOI: 10.1145/3459637.3482306.

[37] L. C. Sun, J. Wang, P. S. Yu, B. Li. Adversarial attack and defense on graph data: A survey, [Online], Available: https://arxiv.org/abs/1812.10528, 2018.

[38] B. Z. Wu, J. T. Li, C. B. Hou, G. J. Fu, Y. T. Bian, L. Chen, J. Z. Huang. Recent advances in reliable deep graph learning: Adversarial attack, inherent noise, and distribution shift, [Online], Available: https://arxiv.org/abs/2202.07114, 2022.

[39] Y. Q. Zhu, W. Z. Xu, J. H. Zhang, Q. Liu, S. Wu, L. Wang. Deep graph structure learning for robust representations: A survey, [Online], Available: https://arxiv.org/abs/2103.03036v1, 2021.

[40] H. C. Liu, Y. Q. Wang, W. Q. Fan, X. R. Liu, Y. X. Li, S. Jain, Y. H. Liu, A. Jain, J. L. Tang. Trustworthy AI: A computational perspective. *ACM Transactions on Intelligent Systems and Technology*, vol. 14, no. 1, Article number 4, 2023. DOI: 10.1145/3546872.

[41] J. Zhou, G. Q. Cui, S. D. Hu, Z. Y. Zhang, C. Yang, Z. Y. Liu, L. F. Wang, C. C. Li, M. S. Sun. Graph neural networks: A review of methods and applications. *AI Open*, vol. 1, pp. 57–81, 2020. DOI: 10.1016/j.aiopen.2021.01.001.

[42] T. X. Zhao, X. Zhang, S. H. Wang. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, ACM, Israel, pp. 833–841, 2021. DOI: 10.1145/3437963.3441720.

[43] B. B. Xu, H. W. Shen, B. J. Sun, R. An, Q. Cao, X. Q. Cheng. Towards consumer loan fraud detection: Graph neural networks with role-constrained conditional random field. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pp. 4537–4545, 2021. DOI: 10.1609/aaai.v35i5.16582.

[44] F. Yang, K. J. Fan, D. D. Song, H. K. Lin. Graph-based prediction of protein-protein interactions with attributed signed graph embedding. *BMC Bioinformatics*, vol. 21, no. 1, Article number 323, 2020. DOI: 10.1186/s12859-020-03646-8.

[45] X. T. Zhang, H. Liu, Q. M. Li, X. M. Wu. Attributed graph clustering via adaptive graph convolution. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, Macao, China, pp. 4327–4333, 2019. DOI: 10.24963/ijcai.2019/601.

[46] X. F. Wang, J. H. Li, L. Yang, H. M. Mi. Unsupervised learning for community detection in attributed networks based on graph convolutional network. *Neurocomputing*, vol. 456, pp. 147–155, 2021. DOI: 10.1016/j.neucom.2021.05.058.

[47] O. Shchur, S. Günnemann. Overlapping community detection with graph neural networks, [Online], Available: https://arxiv.org/abs/1909.12201, 2019.

[48] Z. D. Chen, X. Li, J. Bruna. Supervised community detection with line graph neural networks. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, USA, 2019.

[49] M. Girvan, M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, vol. 99, no. 12, pp. 7821–7826, 2002. DOI: 10.1073/pnas.122653799.

[50] J. O. Garcia, A. Ashourvan, S. Muldoon, J. M. Vettel, D. S. Bassett. Applications of community detection techniques to brain graphs: Algorithmic considerations and implications for neural function. *Proceedings of the IEEE*, vol. 106, no. 5, pp. 846–867, 2018. DOI: 10.1109/JPROC.2017.2786710.

[51] A. Sankar, Y. Z. Liu, J. Yu, N. Shah. Graph neural networks for friend ranking in large-scale social platforms. In *Proceedings of Web Conference 2021*, ACM, Ljubljana, Slovenia, pp. 2535–2546, 2021. DOI: 10.1145/3442381.3450120.

[52] S. Arora. A survey on graph neural networks for knowledge graph completio, [Online], Available: https://arxiv.org/abs/2007.12374, 2020.

[53] V. Duddu, A. Boutet, V. Shejwalkar. Quantifying privacy leakage in graph embedding. In *Proceedings of the 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, ACM, Darmstadt, Germany, pp. 76–85, 2020. DOI: 10.1145/3448891.3448939.

[54] I. E. Olatunji, W. Nejdl, M. Khosla. Membership inference attack on graph neural networks. In *Proceedings of the 3rd IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications*, IEEE, Atlanta, USA, pp. 11–20, 2021. DOI: 10.1109/TPSISA52974.2021.00002.

[55] X. L. He, R. Wen, Y. X. Wu, M. Backes, Y. Shen, Y. Zhang. Node-level membership inference attacks against graph neural networks, [Online], Available: https://arxiv.org/abs/2102.05429, 2021.

[56] B. Wu, X. W. Yang, S. R. Pan, X. L. Yuan. Adapting membership inference attacks to GNN for graph classification: Approaches and implications. In *Proceedings of*

IEEE International Conference on Data Mining, IEEE, Auckland, New Zealand, pp. 1421–1426, 2021. DOI: 10. 1109/ICDM51629.2021.00182.

[57] Z. X. Zhang, Q. Liu, Z. Y. Huang, H. Wang, C. Q. Lu, C. R. Liu, E. H. Chen. GraphMI: Extracting private graph data from graph neural networks. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, Montreal, Canada, pp. 3749–3755, 2021. DOI: 10. 24963/ijcai.2021/516.

[58] B. Wu, X. W. Yang, S. R. Pan, X. L. Yuan. Model extraction attacks on graph neural networks: Taxonomy and realization, [Online], Available: https://arxiv.org/abs/ 2010.12751, 2020.

[59] Y. Shen, X. L. He, Y. F. Han, Y. Zhang. Model stealing attacks against inductive graph neural networks. In *Proceedings of IEEE Symposium on Security and Privacy*, IEEE, San Francisco, USA, pp. 1175–1192, 2022. DOI: 10. 1109/SP46214.2022.9833607.

[60] Y. Li, B Y. Qian, X. L Zhang, H. Liu. Graph neural network-based diagnosis prediction. *Big Data*, vol. 8, no. 5, pp. 379–390, 2020. DOI: 10.1089/big.2020.0070.

[61] Y. H. Long, M. Wu, Y. Liu, Y. Fang, C. K. Kwoh, J. M. Chen, J. W. Luo, X. L. Li. Pre-training graph neural networks for link prediction in biomedical networks. *Bioinformatics*, vol. 38, no. 8, pp. 2254–2262, 2022. DOI: 10. 1093/bioinformatics/btac100.

[62] M. Rigaki, S. Garcia. A survey of privacy attacks in machine learning. *ACM Computing Surveys*, vol. 56, no. 4, pp. 101, 2023. DOI: 10.1145/3624010.

[63] M. Al-Rubaie, J. M. Chang. Privacy-preserving machine learning: Threats and solutions. *IEEE Security & Privacy*, vol. 17, no. 2, pp. 49–58, 2019. DOI: 10.1109/MSEC. 2018.2888775.

[64] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. El Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. Y. He, L. He, Z. Y. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. K. Song, S. U. Stich, Z. T. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Y. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, S. Zhao. Advances and open problems in federated learning, [Online], Available: https://arxiv. org/abs/1912.04977, 2019.

[65] Z. L. Ji, Z. C. Lipton, C. Elkan. Differential privacy and machine learning: a survey and review, [Online], Available: https://arxiv.org/abs/1412.7584, 2014.

[66] M. M. Yang, L. J. Lyu, J. Zhao, T. Q. Zhu, K. Y. Lam. Local differential privacy and its applications: A comprehensive survey, [Online], Available: https://arxiv.org/ abs/2008.03686, 2020.

[67] X. C. Niu, B. F. Li, C. L. Li, R. Xiao, H. C. Sun, H. B. Deng, Z. Z, Chen. A dual heterogeneous graph attention network to improve long-tail performance for shop search in e-commerce. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, USA, pp. 3405–3415, 2020. DOI: 10. 1145/3394486.3403393.

[68] Y. Rong, Y. T Bian, T. Y. Xu, W. Y. Xie, Y. Wei, W. B. Huang, J. Z. Huang. Self-supervised graph transformer on large-scale molecular data. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 1053, 2020.

[69] R. Shokri, M. Stronati, C. Z. Song, V. Shmatikov. Membership inference attacks against machine learning models. In *Proceedings of IEEE Symposium on Security and Privacy*, IEEE, San Jose, USA, pp. 3–18, 2017. DOI: 10. 1109/SP.2017.41.

[70] M. A. Rahman, T. Rahman, R. Laganière, N. Mohammed, Y. Wang. Membership inference attack against differentially private deep learning model. *Transactions on Data Privacy*, vol. 11, no. 1, pp. 61–79, 2018

[71] B. Jayaraman, D. Evans. Evaluating differentially private machine learning in practice. In *Proceedings of the 28th USENIX Security Symposium*, Santa Clara, USA, pp. 1895–1912, 2019.

[72] Y. Chen, L. F. Wu, M. Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 1620, 2020.

[73] I. E. Olatunji, T. Funke, M. Khosla. Releasing graph neural networks with differential privacy guarantees, [Online], Available: https://arxiv.org/abs/2109.08907, 2021.

[74] S. Sajadmanesh, D. Gatica-Perez. Locally private graph neural networks. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, ACM, pp. 2130–2145, 2021. DOI: 10.1145/3460120. 3484565.

[75] D. P. Xu, S. H. Yuan, X. T. Wu, H. Phan. DPNE: Differentially private network embedding. In *Proceedings of the 22nd Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, Springer, Melbourne, Australia, pp. 235–246, 2018. DOI: 10.1007/978-3-319-93037-4_19.

[76] S. J. Zhang, H. Z. Yin, T. Chen, Z. Huang, L. Z. Cui, X. L. Zhang. Graph embedding for recommendation against attribute inference attacks. In *Proceedings of Web Conference 2021*, ACM, Ljubljana, Slovenia, pp. 3002–3014, 2021. DOI: 10.1145/3442381.3449813.

[77] C. Y. He, E. Ceyani, K. Balasubramanian, M. Annavaram, S. Avestimehr. SpreadGNN: Serverless multi-task federated learning for graph neural networks, [Online], Available: https://arxiv.org/abs/2106.02743, 2021.

[78] Y. Pei, R. X. Mao, Y. Liu, C. R. Chen, S. F. Xu, F. Qiang, Blue Elephant Tech. Decentralized federated graph neural networks. In *Proceedings of International Workshop on Federated and Transfer Learning for Data Sparsity and Confidentiality in Conjunction with IJCAI*, 2021.

[79] H. Xie, J. Ma, L. Xiong, C. Yang. Federated graph classification over non-IID graphs. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, Vancouver, Canada, pp. 18839–18852, 2021.

[80] B. H. Wang, A. Li, H. Li, Y. R. Chen. GraphFL: A federated learning framework for semi-supervised node classification on graphs, [Online], Available: https://arxiv.org/ abs/2012.04187, 2020.

[81] L. F. Zheng, J. Zhou, C. C. Chen, B. Z. Wu, L. Wang, B. Y. Zhang. ASFGNN: Automated separated-federated graph neural network. *Peer-to-Peer Networking and Applications*, vol. 14, no. 3, pp. 1692–1704, 2021. DOI: 10.1007/s12083-021-01074-w.

[82] J. Zhou, C. C. Chen, L. F. Zheng, H. W. Wu, J. Wu, X. L. Zheng, B. Z. Wu, Z. Q. Liu, L. Wang. Vertically federated graph neural network for privacy-preserving node classification, [Online], Available: https://arxiv.org/abs/2005.11903, 2020.

[83] Z. W. Liu, L. W. Yang, Z. W. Fan, H. Peng, P. S. Yu. Federated social recommendation with graph neural network. *ACM Transactions on Intelligent Systems and Technolog*, vol. 13, no. 4, Article number 55, 2022. DOI: 10.1145/3501815.

[84] M. Chen, Z. K. Zhang, T. H. Wang, M. Backes, M. Humbert, Y. Zhang. Graph unlearning. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, ACM, Los Angeles, USA, pp. 499–513, 2022. DOI: 10.1145/3548606.3559352.

[85] K. Wu, J. Shen, Y. Ning, T. Wang, W. H. Wang. Certified edge unlearning for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ACM, Long Beach, USA, pp. 2606–2617, 2023. DOI: 10.1145/3580305.3599271.

[86] J. C. Wu, Y. Yang, Y. C. Qian, Y. D. Sui, X. Wang, X. N. He. Gif: A general graph unlearning strategy via influence function. In *Proceedings of ACM Web Conference 2023*, ACM, Austin, USA, pp. 651–661, 2023. DOI: 10.1145/3543507.3583521.

[87] E. Chien, C. Pan, O. Milenkovic. Efficient model updates for approximate unlearning of graph-structured data. In *Proceedings of the 11th International Conference on Learning Representations*, Kigali, Rwanda, 2022.

[88] K. Y. Li, G. C. Luo, Y. Ye, W. Li, S. H. Ji, Z. P. Cai. Adversarial privacy-preserving graph embedding against inference attack. *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6904–6915, 2021. DOI: 10.1109/JIOT.2020.3036583.

[89] B. H. Wang, J. Y. Guo, A. Li, Y. R. Chen, H. Li. Privacy-preserving representation learning on graphs: A mutual information perspective. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ACM, Singapore, pp. 1667–1676, 2021. DOI: 10.1145/3447548.3467273.

[90] J. Xu, S. Koffas, O. Ersoy, S. Picek. Watermarking graph neural networks based on backdoor attacks. In *Proceedings of the 8th IEEE European Symposium on Security and Privacy*, IEEE, Delft, Netherlands, pp. 1179–1197, 2023. DOI: 10.1109/EuroSP57164.2023.00072.

[91] X. Y. Zhao, H. Z. Wu, X. P. Zhang. Watermarking graph neural networks by random graphs. In *Proceedings of the 9th International Symposium on Digital Forensics and Security*, IEEE, Elazig, Turkey, 2021. DOI: 10.1109/ISDFS52919.2021.9486352.

[92] A. Waheed, V. Duddu, N. Asokan. GrOVe: Ownership verification of graph neural networks using embeddings, [Online], Available: https://arxiv.org/abs/2304.08566, 2023.

[93] E. Y. Dai, L. M. Cui, Z. Y. Wang, X. F. Tang, Y. H.

Wang, M. Cheng, B. Yin, S. H. Wang. A unified framework of graph information bottleneck for robustness and membership privacy. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ACM, Long Beach, USA, pp. 368–379, 2023. DOI: 10.1145/3580305.3599248.

[94] C. Dwork, F. McSherry, K. Nissim, A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Theory of Cryptography Conference*, Springer, New York, USA, pp. 265–284, 2006. DOI: 10.1007/11681878_14.

[95] C. Dwork, A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014. DOI: 10.1561/0400000042.

[96] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, L. Zhang. Deep learning with differential privacy. In *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, ACM, Vienna, Austria, pp. 308–318, 2016. DOI: 10.1145/2976749.2978318.

[97] P. C. M. Arachchige, P. Bertok, I. Khalil, D. X. Liu, S. Camtepe, M. Atiquzzaman. Local differential privacy for deep learning. *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 5827–5842, 2020. DOI: 10.1109/JIOT.2019.2952146.

[98] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France, 2017.

[99] S. Zhang, W. W. Ni. Graph embedding matrix sharing with differential privacy. *IEEE Access*, vol. 7, pp. 89390–89399, 2019. DOI: 10.1109/ACCESS.2019.2927365.

[100] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, A. Smith. What can we learn privately? *SIAM Journal on Computing*, vol. 40, no. 3, pp. 793–826, 2011. DOI: 10.1137/090756090.

[101] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, D. Bacon. Federated learning: Strategies for improving communication efficiency, [Online], Available: https://arxiv.org/abs/1610.05492, 2016.

[102] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. Y. Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial intelligence and statistics*, Fort Lauderdale, USA, pp. 1273–1282, 2017.

[103] Q. Yang, Y. Liu, T. J. Chen, Y. X. Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, Article number 12, 2019. DOI: 10.1145/3298981.

[104] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. R. Jia, A. Travers, B. W.Zhang, D. Lie, N. Papernot. Machine unlearning. In *Proceedings of IEEE Symposium on Security and Privacy*, IEEE, San Francisco, USA, pp. 141–159, 2021. DOI: 10.1109/SP40001.2021.00019.

[105] C. Guo, T. Goldstein, A. Hannun, L. Van Der Maaten. Certified data removal from machine learning models. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 3832–3842, 2020.

[106] J. Z. Qiu, Q. B. Chen, Y. X. Dong, J. Zhang, H. X. Yang,

M. Ding, K. S. Wang. J. Tang. GCC: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, USA, pp. 1150–1160, 2020. DOI: 10.1145/3394486.3403168.

[107] Y. C. Sun, T. P. Liu, P. H. Hu, Q. Liao, S. L. Ji, N. H. Yu, D. K. Guo, L. Liu. Deep intellectual property: A survey, [Online], Available: https://arxiv.org/abs/2304.14613, 2023.

[108] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, T. Eliassi-Rad. Collective classification in network data. *AI magazine*, vol. 29, no. 3, pp. 93–106, 2008. DOI: 10.1609/aimag.v29i3.2157.

[109] S. R. Pan, J. Wu, X. Q. Zhu, C. Q. Zhang, Y. Wang. Tri-party deep network representation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, New York, USA, pp. 1895–1901, 2016.

[110] J. McAuley, J. Leskovec. Learning to discover social circles in ego networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, Lake Tahoe, USA, pp. 539–547, 2012.

[111] J. McAuley, J. Leskovec. Image labeling on a network: Using social-network metadata for image classification. In *Proceedings of the 12th European Conference on Computer Vision*, Springer, Florence, Italy, pp. 828–841, 2012. DOI: 10.1007/978-3-642-33765-9_59.

[112] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann. Pitfalls of graph neural network evaluation, [Online], Available: https://arxiv.org/abs/1811.05868, 2018.

[113] X. Wang, H. Y. Ji, C. Shi, B. Wang, Y. F. Ye, P. Cui, P. S. Yu. Heterogeneous graph attention network. In *Proceedings of World Wide Web Conference*, ACM, San Francisco, USA, pp. 2022–2032, 2019. DOI: 10.1145/3308558.3313562.

[114] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, M. Neumann. TUDataset: A collection of benchmark datasets for learning with graphs, [Online], Available: https://arxiv.org/abs/2007.08663, v1 2020.

[115] K. D. Xu, H. G. Chen, S. J. Liu, P. Y. Chen, T. W. Weng, M. Y. Hong, X. Lin. Topology attack and defense for graph neural networks: An optimization perspective. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, Macao, China, pp. 3961–3967, 2019. DOI: 10.24963/ijcai.2019/550.

[116] J. Y. Chen, Y. Y. Wu, X. H. Xu, Y. X. Chen, H. B. Zheng, Q. Xuan. Fast gradient attack on network embedding, [Online], Available: https://arxiv.org/abs/1809.02797, 2018.

[117] H. J. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, L. Song. Adversarial attack on graph structured data. In *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, pp. 1115–1124, 2018.

[118] S. Geisler, T. Schmidt, H. Şirin, D. Zügner, A. Bojchevski, S. Günnemann. Robustness of graph neural networks at scale. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, Vancouver, Canada, pp. 7637–7649, 2021.

[119] Y. Ma, S. H. Wang, T. Derr, L. F. Wu, J. L. Tang. Graph adversarial attack via rewiring. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ACM, Singapore, pp. 1161–1169, 2021. DOI: 10.1145/3447548.3467416.

[120] D. Zügner, A. Akbarnejad, S. Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, London, UK, pp. 2847–2856, 2018. DOI: 10.1145/3219819.3220078.

[121] D. Zügner, S. Günnemann. Adversarial attacks on graph neural networks via meta learning. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, USA, 2019.

[122] J. H. Wang, M. N. Luo, F. Suya, J. D. Li, Z. J. Yang, Q. H. Zheng. Scalable attack on graph data by injecting vicious nodes. *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1363–1389, 2020. DOI: 10.1007/s10618-020-00696-7.

[123] A. Bojchevski, S. Günnemann. Adversarial attacks on node embeddings via graph poisoning. In *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, USA, pp. 695–704, 2019.

[124] H. Chang, Y. Rong, T. Y. Xu, W. B. Huang, H. L. Zhang, P. Cui, W. W. Zhu, J. Z. Huang. A restricted black-box adversarial framework towards attacking graph embedding models. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, AAAI, New York, USA, pp. 3389–3396, 2020. DOI: 10.1609/aaai.v34i04.5741.

[125] S. C. Tao, Q. Cao, H. W. Shen, J. J. Huang, Y. F. Wu, X. Q. Cheng. Single node injection attack against graph neural networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ACM, Queensland, Australia, pp. 1794–1803, 2021. DOI: 10.1145/3459637.3482393.

[126] J. Q. Ma, S. R. Ding, Q. Z. Mei. Towards more practical adversarial attacks on graph neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 399, 2020.

[127] Z. H. Xi, R. Pang, S. L. Ji, T. Wang. Graph backdoor. In *Proceedings of the 30th USENIX Security Symposium*, pp. 1523–1540, 2021.

[128] Z. X. Zhang, J. Y. Jia, B. H. Wang, N. Z. Gong. Backdoor attacks to graph neural networks. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, ACM, Spain, pp. 15–26, 2021. DOI: 10.1145/3450569.3463560.

[129] J. T. Li, T. Xie, L. Chen, F. F. Xie, X. N. He, Z. B. Zheng. Adversarial attack on large scale graph. *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 82–95, 2023. DOI: 10.1109/TKDE.2021.3078755.

[130] Z. N. Hu, Y. X. Dong, K. S. Wang, K. W. Chang, Y. Z. Sun. GPT-GNN: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, USA, pp. 1857–1867, 2020. DOI: 10.1145/3394486.3403237.

[131] D. Ahmedt-Aristizabal, M. A. Armin, S. Denman, C. Fookes, L. Petersson. Graph-based deep learning for medical diagnosis and analysis: Past, present and future. *Sensors*, vol. 21, no. 14, Article number 4758, 2021. DOI: 10.3390/s21144758.

[132] W. H. Hu, B. W. Liu, J. Gomes, M. Zitnik, P. Liang, V.

Pande, J. Leskovec. Strategies for pre-training graph neural networks. In *Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

[133] L. Chen, J. T. Li, J. Y. Peng, T. Xie, Z. X. Cao, K. Xu, X. N. He, Z. B Zheng. A survey of adversarial learning on graphs, [Online], Available: https://arxiv.org/abs/2003.05730, 2020.

[134] F. Mujkanovic, S. Geisler, S. Günnemann, A. Bojchevski. Are defenses for graph neural networks robust? In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, New Orleans, USA, Article number 651, 2022.

[135] E. Y. Dai, M. H. Lin, X. Zhang, S. H. Wang. Unnoticeable backdoor attacks on graph neural networks. In *Proceedings of ACM Web Conference 2023*, ACM, Austin, USA, pp. 2263–2273, 2023. DOI: 10.1145/3543507.3583392.

[136] X. Zou, Q. K. Zheng, Y. X. Dong, X. Y. Guan, E. Kharlamov, J. L. Lu, J. Tang. TDGIA: Effective injection attacks on graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ACM, Singapore, pp. 2461–2471, 2021. DOI: 10.1145/3447548.3467314.

[137] Y. Q. Chen, H. Yang, Y. G. Zhang, K. L. Ma, T. L. Liu, B. Han, J. Cheng. Understanding and improving graph injection attack by promoting unnoticeability. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

[138] J. Xu, R. Wang, S. Koffas, K. T. Liang, S. Picek. More is better (mostly): On the backdoor attacks in federated graph neural networks. In *Proceedings of the 38th Annual Computer Security Applications Conference*, ACM, Austin, USA, pp. 684-698, 2022. DOI: 10.1145/3564625.3567999.

[139] I. J. Goodfellow, J. Shlens, C. Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, USA, 2015. DOI: 10.48550/arXiv.1412.6572.

[140] Q. Y. Dai, X Shen, L. Zhang, Q. Li, D. Wang. Adversarial training methods for network embedding. In *Proceedings of World Wide Web Conference*, ACM, San Francisco, USA, pp. 329–339, 2019. DOI: 10.1145/3308558.3313445.

[141] Z. J. Deng, Y. P. Dong, J. Zhu. Batch virtual adversarial training for graph convolutional networks. *AI Open*, vol. 4, pp. 73–79, 2023. DOI: 10.1016/j.aiopen.2023.08.007.

[142] F. L. Feng, X. N. He, J. Tang, T. S. Chua. Graph adversarial training: Dynamically regularizing based on graph structure. *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 6, pp. 2493–3504, 2021. DOI: 10.1109/TKDE.2019.2957786.

[143] J. H. Wang, M. N. Luo, J. D. Li, Z. Q. Liu, J. Zhou, Q. H. Zheng. Robust unsupervised graph representation learning via mutual information maximization, [Online], Available: https://arxiv.org/abs/2201.08557, 2022.

[144] L. Gosch, S. Geisler, D. Sturm, B. Charpentier, D. Zügner, S. Günnemann. Adversarial training for graph neural networks, [Online], Available: https://arxiv.org/abs/2306.15427v1, 2023.

[145] J. Y. Chen, Y. Y. Wu, X. Lin, Q. Xuan. Can adversarial network attack be defended? [Online], Available: https://arxiv.org/abs/1903.05994, 2019.

[146] X. Y. Wang, X. Q. Liu, C. J. Hsieh. GraphDefense: Towards robust graph convolutional networks, [Online], Available: https://arxiv.org/abs/1911.04429, 2019.

[147] D. Zügner, S. Günnemann. Certifiable robustness and robust training for graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, Anchorage, USA, pp. 246–256, 2019. DOI: 10.1145/3292500.3330905.

[148] B. H. Wang, J. Y. Jia, X. Y. Cao, N. Z. Gong. Certified robustness of graph neural networks against adversarial structural perturbation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ACM, Singapore, pp. 1645–1653, 2021. DOI: 10.1145/3447548.3467295.

[149] H. W. Jin, Z. Shi, A. Peruri, X. H. Zhang. Certified robustness of graph convolution networks for graph classification under topological attacks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 709, 2020.

[150] D. Zügner, S. Günnemann. Certifiable robustness of graph convolutional networks under structure perturbations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, USA, pp. 1656–1665, 2020. DOI: 10.1145/3394486.3403217.

[151] A. Bojchevski, S. Günnemann. Certifiable robustness to graph perturbations. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 747, 2019.

[152] J. Klicpera, A. Bojchevski, S. Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, USA, 2019.

[153] Y. Scholten, J. Schuchardt, S. Geisler, A. Bojchevski, S. Günnemann. Randomized message-interception smoothing: Gray-box certificates for graph neural networks. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, New Orleans, USA, Article number 2402, 2022.

[154] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, E. 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*, ACM, Houston, USA, pp. 169–177, 2020. DOI: 10.1145/3336191.3371789.

[155] D. S. Luo, W. Cheng, W. C. Yu, B. Zong, J. C. Ni, H. F. Chen, X. Zhang. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, ACM, Israel, pp. 779–787, 2021. DOI: 10.1145/3437963.3441734.

[156] L. Wang, W. C. Yu, W. Wang, W. Cheng, W. Zhang, H. Y. Zha, X. F. He, H. F. Chen. Learning robust representations with graph denoising policy network. In *Proceed-*

ings of IEEE International Conference on Data Mining, IEEE, Beijing, China, pp. 1378–1383, 2019. DOI: 10.1109/ICDM.2019.00177.

[157] X. F. Tang, Y. D. Li, Y. W. Sun, H. X. Yao, P. Mitra, S. H. Wang. Transferring robustness for graph neural network against poisoning attacks. In Proceedings of the 13th International Conference on Web Search and Data Mining, ACM, Houston, USA, pp. 600–608, 2020. DOI: 10.1145/3336191.3371851.

[158] X. Zhang, M. Zitnik. GNNGUARD: Defending graph neural networks against adversarial attacks. In Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, Canada, Article number 777, 2020.

[159] D. Y. Zhu, Z. W. Zhang, P. Cui, W. W. Zhu. Robust graph convolutional networks against adversarial attacks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ACM, Anchorage, USA, pp. 1399–1407, 2019. DOI: 10.1145/3292500.3330851.

[160] L. Chen, J. T. Li, Q. B. Peng, Y. Liu, Z. B. Zheng, C. Yang. Understanding structural vulnerability in graph convolutional networks. In Proceedings of the 30th International Joint Conference on Artificial Intelligence, Canada, pp. 2249–2255, 2021. DOI: 10.24963/ijcai.2021/310.

[161] D. Kim, A. Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In Proceedings of the 9th International Conference on Learning Representations, Austria, 2021.

[162] Y. N. You, T. L. Chen, Y. D. Sui, T. Chen, Z. Y. Wang, Y. Shen. Graph contrastive learning with augmentations. In Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, Canada, Article number 488, 2020.

[163] W. Jin, T. Derr, Y. Q. Wang, Y. Ma, Z. T. Liu, J. L. Tang. Node similarity preserving graph convolutional networks. In Proceedings of the 14th ACM International Conference on Web Search and Data Mining, ACM, Israel, pp. 148–156, 2021. DOI: 10.1145/3437963.3441735.

[164] J. Y. Guo, S. Y. Li, Y. Zhao, Y. Zhang. Learning robust representation through graph adversarial contrastive learning. In Proceedings of the 27th International Conference on Database Systems for Advanced Applications, Springer, pp. 682–697, 2022. DOI: 10.1007/978-3-031-00123-9_54.

[165] S. Y. Feng, B. Y. Jing, Y. D. Zhu, H. H. Tong. Adversarial graph contrastive learning with information regularization. In Proceedings of ACM Web Conference 2022, ACM, pp. 1362–1371, 2022. DOI: 10.1145/3485447.3512183.

[166] L. Lin, J. H. Chen, H. N. Wang. Spectral augmentation for self-supervised learning on graphs. In Proceedings of the 11th International Conference on Learning Representations, Kigali, Rwanda, 2023.

[167] H. T. Tang, G. X. Ma, Y. R. Chen, L. Guo, W. Wang, B. Zeng, L. Zhan. Adversarial attack on hierarchical graph pooling neural networks, [Online], Available: https://arxiv.org/abs/2005.11560, 2020.

[168] H. T. Zhang, T. H. Zheng, J. Gao, C. L. Miao, L. Su, Y. L. Li, K. Ren. Data poisoning attack against knowledge graph embedding. In Proceedings of the 28th Internation-

al Joint Conference on Artificial Intelligence, Macao, China, pp. 4853–4859, 2019. DOI: 10.24963/ijcai.2019/674.

[169] H. W. Wang, F. Z. Zhang, X. Xie, M. Y. Guo. DKN: Deep knowledge-aware network for news recommendation. In Proceedings of World Wide Web Conference, International World Wide Web Conferences Steering Committee, Lyon, France, pp. 1835–1844, 2018. DOI: 10.1145/3178876.3186175.

[170] C. H. Deng, X. Y. Li, Z. Feng, Z. R. Zhang. GARNET: Reduced-rank topology learning for robust and scalable graph neural networks. In Proceedings of the 1st Learning on Graphs Conference, Article number 3, 2022.

[171] M. M. Zhang, X. Wang, M. Q. Zhu, C. Shi, Z. Q. Zhang, J. Zhou. Robust heterogeneous graph neural networks against adversarial attacks. In Proceedings of the 36th AAAI Conference on Artificial Intelligence, AAAI, pp. 4363–4370, 2022. DOI: 10.1609/aaai.v36i4.20357.

[172] M. M. Zhang, L. M. Hu, C. Shi, X. Wang. Adversarial label-flipping attack and defense for graph neural networks. In Proceedings of IEEE International Conference on Data Mining, IEEE, Sorrento, Italy, pp. 791–800, 2020. DOI: 10.1109/ICDM50108.2020.00088.

[173] Y. Y. Li, J. Yin, L. Chen. Unified robust training for graph neural networks against label noise. In Proceedings of the 25th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, Springer, pp. 528–540, 2021. DOI: 10.1007/978-3-030-75762-5_42.

[174] E. Y. Dai, C. Aggarwal, S. H. Wang. NRGNN: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, ACM, Singapore, pp. 227–236, 2021. DOI: 10.1145/3447548.3467364.

[175] T. Rahman, B. Surma, M. Backes, Y. Zhang. Fairwalk: Towards fair graph embedding. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, Macao, China, pp. 3289–3295, 2019. DOI: 10.24963/ijcai.2019/456.

[176] A. A. Stoica, C. Riederer, A. Chaintreau. Algorithmic glass ceiling in social networks: The effects of social recommendations on network diversity. In Proceedings of World Wide Web Conference, International World Wide Web Conferences Steering Committee, Lyon, France, pp. 923–932, 2018. DOI: 10.1145/3178876.3186140.

[177] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, A. Galstyan. A survey on bias and fairness in machine learning. ACM Computing Surveys, vol. 54, no. 6, Article number 115, 2021. DOI: 10.1145/3457607.

[178] A. Olteanu, C. Castillo, F. Diaz, E. Kıcıman. Social data: Biases, methodological pitfalls, and ethical boundaries. Frontiers in Big Data, vol. 2, Article number 13, 2019. DOI: 10.3389/fdata.2019.00013.

[179] H. Suresh, J. V. Guttag. A framework for understanding unintended consequences of machine learning, [Online], Available: https://arxiv.org/abs/1901.10002, 2019.

[180] Y. S. Dong, N. H. Liu, B. Jalaian, J. D. Li. EDITS: Modeling and mitigating data bias for graph neural networks. In Proceedings of ACM Web Conference 2022, ACM, pp. 1259–1269, 2022. DOI: 10.1145/3485447.3512173.

[181] Y. X. Dong, O. Lizardo, N. V. Chawla. Do the young live

in a "smaller world" than the old? Age-specific degrees of separation in a large-scale mobile communication network, [Online], Available: https://arxiv.org/abs/1606.07556, 2016.

[182] X. F. Tang, H. X. Yao, Y. W. Sun, Y. Q. Wang, J. L. Tang, C. Aggarwal, P. Mitra, S. H. Wang. Investigating and mitigating degree-related biases in graph convoltuional networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, ACM, Ireland, pp. 1435–1444, 2020. DOI: 10.1145/3340531.3411872.

[183] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, R. Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ACM, Cambridge, USA, pp. 214–226, 2012. DOI: 10.1145/2090236.2090255.

[184] F. Locatello, G. Abbati, T. Rainforth, S. Bauer, B. Schölkopf, O. Bachem. On the fairness of disentangled representations. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 1309, 2019.

[185] M. Hardt, E. Price, N. Srebro. Equality of opportunity in supervised learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, Barcelona, Spain, pp. 3323–3331, 2016.

[186] P. Z. Li, Y. F. Wang, H. Zhao, P. Y. Hong, H. F. Liu. On dyadic fairness: Exploring and mitigating bias in graph connections. In *Proceedings of the 9th International Conference on Learning Representations*, Austria, 2021.

[187] M. Kusner, J. Loftus, C. Russell, R. Silva. Counterfactual fairness. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Long Beach, USA, pp. 4069–4079, 2017.

[188] J. Kang, J. R. He, R. Maciejewski, H. H. Tong. InFoRM: Individual fairness on graph mining. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, USA, pp. 379–389, 2020. DOI: 10.1145/3394486.3403080.

[189] C. Agarwal, H. Lakkaraju, M. Zitnik. Towards a unified framework for fair and stable graph representation learning. In *Proceedings of the 37th Conference on Uncertainty in Artificial Intelligence*, pp. 2114–2124, 2021.

[190] I. Spinelli, S. Scardapane, A. Hussain, A. Uncini. Biased edge dropout for enhancing fairness in graph representation learning, [Online], Available: https://arxiv.org/abs/2104.14210v1, 2021.

[191] A. Bose, W. Hamilton. Compositional fairness constraints for graph embeddings. In *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, USA, pp. 715–724, 2019.

[192] F. Masrour, T. Wilson, H. Yan, P. N. Tan, A. Esfahanian. Bursting the filter bubble: Fairness-aware network link prediction. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, AAAI, New York, USA, pp. 841–848, 2020. DOI: 10.1609/aaai.v34i01.5429.

[193] N. Wang, L. Lin, J. D. Li, H. N. Wang. Unbiased graph embedding with biased graph observations. In *Proceedings of ACM Web Conference 2022*, ACM, Lyon, France, pp. 1423–1433, 2022. DOI: 10.1145/3485447.3512189.

[194] Y. S. Dong, J. Kang, H. H. Tong, J. D. Li. Individual fairness for graph neural networks: A ranking based approach. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ACM, Singapore, pp. 300–310, 2021. DOI: 10.1145/3447548.3467266.

[195] Ö. D. Köse, Y. N. Shen. Fairness-aware node representation learning, [Online], Available: https://arxiv.org/abs/2106.05391, 2021.

[196] A. Khajehnejad, M. Khajehnejad, M. Babaei, K. P. Gummadi, A. Weller, B. Mirzasoleiman. CrossWalk: Fairness-enhanced node representation learning. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, AAAI, pp. 11963–11970, 2022. DOI: 10.1609/aaai.v36i11.21454.

[197] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, Cambridge, USA, pp. 2672–2680, 2014.

[198] A. Beutel, J. L. Chen, Z. Zhao, E. H. Chi. Data decisions and theoretical implications when adversarially learning fair representations, [Online], Available: https://arxiv.org/abs/1707.00075, 2017.

[199] H. Edwards, A. Storkey. Censoring representations with an adversary. In *Proceedings of the 4th International Conference on Learning Representations*, San Juan, Puerto Rico, 2016. DOI: 10.48550/arXiv.1511.05897.

[200] J. C. Liao, C. Huang, P. Kairouz, L. Sankar. Learning generative adversarial representations (GAP) under fairness and censoring constraints, [Online], Available: https://arxiv.org/abs/1910.00411, 2019.

[201] D. Madras, E. Creager, T. Pitassi, R. Zemel. Learning adversarially fair and transferable representations. In *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, pp. 3384–3393, 2018.

[202] M. Arjovsky, S. Chintala, L. Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, pp. 214–223, 2017.

[203] Z. M. Guo, T. Xiao, C. Aggarwal, H. Liu, S. H. Wang. Counterfactual learning on graphs: A survey, [Online], Available: https://arxiv.org/abs/2304.01391, 2023.

[204] H. S. Zhu, G. J. Fu, Z. M. Guo, Z. W. Zhang, T. Xiao, S. H. Wang. Fairness-aware message passing for graph neural networks, [Online], Available: https://arxiv.org/abs/2306.11132, 2023.

[205] T. N. Kipf, M. Welling. Variational graph auto-encoders, [Online], Available: https://arxiv.org/abs/1611.07308, 2016.

[206] Z. M. Guo, J. L. Li, T. Xiao, Y. Ma, S. H. Wang. Towards fair graph neural networks via graph counterfactual. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, ACM, Birmingham, UK, pp. 669–678, 2023. DOI: 10.1145/3583780.3615092.

[207] Y. Wang, Y. Y. Zhao, Y. S. Dong, H. Y. Chen, J. D. Li, T. Derr. Improving fairness in graph neural networks via mitigating sensitive attribute leakage. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ACM, Washington DC, USA, pp. 1938–1948, 2022. DOI: 10.1145/3534678.3539404.

[208] O. D. Kose, Y. N. Shen. Demystifying and mitigating bias for node representation learning. *IEEE Transactions on Neural Networks and Learning Systems*, [Online], Available: https://ieeexplore.ieee.org/abstract/document/10103678. 2023.

[209] H. Y. Ling, Z. M. Jiang, Y. Z. Luo, S. W. Ji, N. Zou. Learning fair graph representations via automated data augmentations. In *Proceedings of the 11th International Conference on Learning Representations*, Kigali, Rwanda, 2023.

[210] Y. S. Dong, S. Wang, Y. Wang, T. Derr, J. D. Li. On structural explanation of bias in graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ACM, Washington DC, USA, pp. 316–326, 2022. DOI: 10.1145/3534678.3539319.

[211] Y. S. Dong, S. Wang, J. Ma, N. H. Liu, J. D. Li. Interpreting unfairness in graph neural networks via training node attribution. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, AAAI, Washington DC, USA, pp. 7441–7449, 2023. DOI: 10.1609/aaai.v37i6.25905.

[212] J. Ma, R. C. Guo, M. T. Wan, L. Q. Yang, A. D. Zhang, J. D. Li. Learning fair node representations with graph counterfactual fairness. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, ACM, USA, pp. 695–703, 2022. DOI: 10.1145/3488560.3498391.

[213] J. Palowitch, B. Perozzi. MONET: Debiasing graph embeddings via the metadata-orthogonal training unit, [Online], Available: https://arxiv.org/abs/1909.11793, 2019.

[214] A. Asuncion, D. Newman. UCI machine learning repository, 2007.

[215] K. L. Jordan, T. L. Freiburger. The effect of race/ethnicity on sentencing: Examining sentence type, jail length, and prison length. *Journal of Ethnicity in Criminal Justice*, vol. 13, no. 3, pp. 179–196, 2015. DOI: 10.1080/15377938.2014.984045.

[216] F. M. Harper, J. A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, Article number 19, 2015. DOI: 10.1145/2827872.

[217] L. A. Adamic, N. Glance. The political blogosphere and the 2004 U.S. election: Divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, ACM, Chicago, USA, pp. 36-43, 2005. DOI: 10.1145/1134271.1134277.

[218] M. Babaei, P. Grabowicz, I. Valera, K. P. Gummadi, M. Gomez-Rodriguez. On the efficiency of the information networks in social media. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*, ACM, San Francisco, USA, pp. 83–92, 2016. DOI: 10.1145/2835776.2835826.

[219] M. Cha, H. Haddadi, F. Benevenuto, P. K. Gummadi. Measuring user influence in twitter: The million follower fallacy. In *Proceedings of the 4th International Conference on Weblogs and Social Media*, AAAI Press, Washington DC, USA, 2010. DOI: 10.1609/icwsm.v4i1.14033.

[220] E. Y. Dai, J. Chen. Graph-augmented normalizing flows for anomaly detection of multiple time series. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

[221] T. A. B. Snijders, G. G. Van de Bunt, C. E. G. Steglich. Introduction to stochastic actor-based models for network dynamics. *Social Networks*, vol. 32, no. 1, pp. 44–60, 2010. DOI: 10.1016/j.socnet.2009.02.004.

[222] R. Ying, R. N. He, K. F. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, London, UK, pp. 974–983, 2018. DOI: 10.1145/3219819.3219890.

[223] T. Liang, G. X. Zeng, Q. W. Zhong, J. F. Chi, J. H. Feng, X. Ao, J. Y. Tang. Credit risk and limits forecasting in E-commerce consumer lending service via multi-view-aware mixture-of-experts nets. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, ACM, Israel, pp. 229–237, 2021. DOI: 10.1145/3437963.3441743.

[224] D. W. Cheng, Y. Tu, Z. W. Ma, Z. B. Niu, L. Q. Zhang. Risk assessment for networked-guarantee loans using high-order graph attention representation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, Macao, China, pp. 5822–5828, 2019. DOI: 10.24963/ijcai.2019/807.

[225] S. X. Rao, S. Zhang, Z. C. Han, Z. T. Zhang, W. Min, Z. Y. Chen, Y. N. Shan, Y. Zhao, C Zhang. xFraud: Explainable fraud transaction detection on heterogeneous graphs, [Online], Available: https://arxiv.org/abs/2011.12193, 2020.

[226] M. H. Van, W. Du, X. T. Wu, A. D. Lu. Poisoning attacks on fair machine learning. In *Proceedings of the 27th International Conference on Database Systems for Advanced Applications*, Springer, pp. 370–386, 2022. DOI: 10.1007/978-3-031-00123-9_30.

[227] Z. N. Hu, Y. X. Dong, K. A. Wang, Y. Z. Sun. Heterogeneous graph transformer. In *Proceedings of Web Conference 2020*, ACM, Taipei, China, pp. 2704–2710, 2020. DOI: 10.1145/3366423.3380027.

[228] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling. Modeling relational data with graph convolutional networks. In *Proceedings of the 15th International Conference on the Semantic Web*, Springer, Heraklion, Greece, pp. 593–607, 2018. DOI: 10.1007/978-3-319-93417-4_38.

[229] Z. Q. Zeng, R. Islam, K. N. Keya, J. Foulds, Y. Q. Song, S. M. Pan. Fair representation learning for heterogeneous information networks. In *Proceedings of the 15th International AAAI Conference on Web and Social Media*, AAAI, pp. 877–887, 2021. DOI: 10.1609/icwsm.v15i1.18111.

[230] Y. R. Shrestha, Y. J. Yang. Fairness in algorithmic decision-making: Applications in multi-winner voting, machine learning, and recommender systems. *Algorithms*, vol. 12, no. 9, Article number 199, 2019. DOI: 10.3390/a12090199.

[231] T. Hashimoto, M. Srivastava, H. Namkoong, P. Liang. Fairness without demographics in repeated loss minimization. In *Proceedings of the 35th International Conference on Machine Learning*, Stockholm, Sweden, pp. 1929–1938, 2018.

[232] T. X. Zhao, E. Y. Dai, K. Shu, S. H. Wang. You can still achieve fairness without sensitive attributes: Exploring biases in non-sensitive features, [Online], Available: ht-

tps://arxiv.org/abs/2104.14537, 2021.

[233] P. Lahoti, A. Beutel, J. L. Chen, K. Lee, F. Prost, N. Thain, X. Z. Wang, E. H. Chi. Fairness without demographics through adversarially reweighted learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 62, 2020.

[234] Q. S. Zhang, S. C. Zhu. Visual interpretability for deep learning: A survey. *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 1, pp. 27–39, 2018. DOI: 10.1631/FITEE.1700808.

[235] M. Nauta, J. Trienes, S. Pathak, E. Nguyen, M. Peters, Y. Schmitt, J. Schlötterer, M. van Keulen, C. Seifert. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable AI. *ACM Computing Surveys*, vol. 55, no. 13s, Article number 295, 2023. DOI: 10.1145/3583558.

[236] S. Sharma, J. Henderson, J. Ghosh. CERTIFAI: A common framework to provide explanations and analyse the fairness and robustness of black-box models. In *Proceedings of AAAI/ACM Conference on AI, Ethics, and Society*, ACM, New York, USA, pp. 166–172, 2020. DOI: 10.1145/3375627.3375812.

[237] H. Baniecki, W. Kretowicz, P. Piątyszek, J. Wisniewski, P. Biecek. dalex: Responsible machine learning with interactive explainability and fairness in python. *Journal of Machine Learning Research*, vol. 22, no. 1, Article number 214, 2021

[238] H. Yuan, H. Y. Yu, S. R. Gui, S. W. Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5782–5799, 2023. DOI: 10.1109/TPAMI.2022.3204236.

[239] Z. X. Zhang, Q. Liu, H. Wang, C. Q. Lu, C. Lee. Prot-GNN: Towards self-explaining graph neural networks. In *Proceedings of the 36th AAAI/ACM Conference on AI, Ethics, and Society*, AAAI, pp. 9127–9135, 2022. DOI: 10.1609/aaai.v36i8.20898.

[240] L. Faber, A. K. Moghaddam, R. Wattenhofer. When comparing to ground truth is wrong: On evaluating gnn explanation methods. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, ACM, Singapore, pp. 332–341, 2021. DOI: 10.1145/3447548.3467283.

[241] B. Mittelstadt, C. Russell, S. Wachter. Explaining explanations in AI. In *Proceedings of Conference on Fairness, Accountability, and Transparency*, ACM, Atlanta, USA, pp. 279–288, 2019. DOI: 10.1145/3287560.3287574.

[242] M. Danilevsky, K. Qian, R. Aharonov, Y. Katsis, B. Kawas, P. Sen. A survey of the state of explainable ai for natural language processing. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, ACL, Suzhou, China, pp. 447–459, 2020.

[243] D. S. Luo, W. Cheng, D. K. Xu, W. C. Yu, B. Zong, H. F. Chen, X. Zhang. Parameterized explainer for graph neural network. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 1646, 2020.

[244] H. Yuan, J. L. Tang, X. Hu, S. W. Ji. XGNN: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ACM, USA, pp. 430–438, 2020. DOI: 10.1145/3394486.3403085.

[245] F. Baldassarre, H. Azizpour. Explainability techniques for graph convolutional networks, [Online], Available: https://arxiv.org/abs/1905.13686, 2019.

[246] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, B. Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences of the United States of America*, vol. 116, no. 44, pp. 22071–22080, 2019. DOI: 10.1073/pnas.1900654116.

[247] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, W. K Wong. Too much, too little, or just right? Ways explanations impact end users′ mental models. In *Proceedings of IEEE Symposium on Visual Languages and Human Centric Computing*, IEEE, San Jose, USA, pp. 3–10, 2013. DOI: 10.1109/VLHCC.2013.6645235.

[248] M. Robnik-Šikonja, M. Bohanec. Perturbation-based explanations of prediction models. *Human and Machine Learning: Visible, Explainable, Trustworthy and Transparent*, J. L. Zhou, F. Chen, Eds., Cham, Switzerland: Springer, pp. 159–175, 2018. DOI: 10.1007/978-3-319-90403-0_9.

[249] D. Alvarez-Melis, T. S. Jaakkola. On the robustness of interpretability methods, [Online], Available: https://arxiv.org/abs/1806.08049, 2018.

[250] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, vol. 267, pp. 1–38, 2019. DOI: 10.1016/j.artint.2018.07.007.

[251] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, B. Kim. Sanity checks for saliency maps. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Montreal, Canada, pp. 9525–9536, 2018.

[252] P. Atanasova, J. G. Simonsen, C. Lioma, I. Augenstein. A diagnostic study of explainability techniques for text classification. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, ACL, pp. 3256–3274, 2020. DOI: 10.18653/v1/2020.emnlp-main.263.

[253] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, B. Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, vol. 51, no. 1, pp. 141–154, 2011. DOI: 10.1016/j.dss.2010.12.003.

[254] M. Bajaj, L. Y. Chu, Z. Y. Xue, J. Pei, L. J. Wang, P. C. H. Lam, Y. Zhang. Robust counterfactual explanations on graph neural networks. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, Vancouver, Canada, pp. 5644–5655, 2021.

[255] S. Verma, V. Boonsanong, M. Hoang, K. E. Hines, J. P. Dickerson, C. Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review, [Online], Available: https://arxiv.org/abs/2010.10596, 2020.

[256] W. Y. Lin, H. Lan, B. C. Li. Generative causal explanations for graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 6666–6679, 2021.

[257] Q. Huang, M. Yamada, Y. Tian, D. Singh, D. W. Yin, Y. Chang. GraphLIME: Local interpretable model explanations for graph neural networks, [Online], Available: https://arxiv.org/abs/2001.06216, 2020.

[258] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, pp. 1263–1272, 2017.

[259] P. E. Pope, S. Kolouri, M. Rostami, C. E. Martin, H. Hoffmann. Explainability methods for graph convolutional neural networks. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Long Beach, USA, pp. 10764–10773, 2019. DOI: 10.1109/CVPR.2019.01103.

[260] R. Schwarzenberg, M. Hübner, D. Harbecke, C. Alt, L. Hennig. Layerwise relevance visualization in convolutional text graph classifiers. In *Proceedings of the 13th Workshop on Graph-Based Methods for Natural Language Processing*, ACL, Hong Kong, China, pp. 58–62, 2019. DOI: 10.18653/v1/D19-5308.

[261] Y. Zhang, D. Defazio, A. Ramesh. ReLEx: A model-agnostic relational model explainer. In *Proceedings of 2021 AAAI/ACM Conference on AI, Ethics, and Society*, ACM, USA, pp. 1042–1049, 2021. DOI: 10.1145/3461702.3462562.

[262] M. N. Vu, M. T. Thai. PGM-explainer: Probabilistic graphical model explanations for graph neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 1025, 2020.

[263] W. T. Xiong, I. Ni'mah, J. M. G. Huesca, W. van Ipenburg, J. Veldsink, M. Pechenizkiy. Looking deeper into deep learning model: Attribution-based explanations of textcnn, [Online], Available: https://arxiv.org/abs/1811.03970, 2018.

[264] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, K. R. Müller. How to explain individual classification decisions. *Journal of Machine Learning Research*, vol. 11, pp. 1803–1831, 2010

[265] M. Sundararajan, A. Taly, Q. Q. Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, pp. 3319–3328, 2017.

[266] Y. Y. Gao, T. Sun, R. Bhatt, D. Z. Yu, S. Hong, L. Zhao. GNES: Learning to explain graph neural networks. In *Proceedings of IEEE International Conference on Data Mining*, IEEE, Auckland, New Zealand, pp. 131–140, 2021. DOI: 10.1109/ICDM51629.2021.00023.

[267] T. Funke, M. Khosla, A. Anand. Hard masking for explaining graph neural networks. In *Proceedings of International Conference on Learning Representations*, Vienna, Austria, 2021.

[268] X. Wang, Y. X. Wu, A. Zhang, X. N. He, T. S. Chua. Causal screening to interpret graph neural networks. In *Proceedings of International Conference on Learning Representations*, Vienna, Austria, 2021.

[269] A. Lucic, M. A. T. Hoeve, G. Tolomei, M. De Rijke, F. Silvestri. CF-GNNExplainer: Counterfactual explanations for graph neural networks. In *Proceedings of the 25th International Conference on Artificial intelligence and Statistics*, pp. 4499–4511, 2022.

[270] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations*, Vancouver, Canada, 2018.

[271] X. Wang, Y. X. Wu, A. Zhang, X. N. He, T. S. Chua. Towards multi-grained explainability for graph neural networks. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, Vancouver, Canada, 2021.

[272] M. S. Schlichtkrull, N. De Cao, I. Titov. Interpreting graph neural networks for nlp with differentiable edge masking. In *Proceedings of the 9th International Conference on Learning Representations*, Austria, 2021.

[273] S. Bach, A. Binder, G. Montavon, F. Klauschen, K. R. Müller, W. Samek. On pixel-wise explanations for nonlinear classifier decisions by layer-wise relevance propagation. *PLoS One*, vol. 10, no. 7, Article number e0130140, 2015. DOI: 10.1371/journal.pone.0130140.

[274] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller. Striving for simplicity: The all convolutional net. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, USA, 2014. DOI: 10.48550/arXiv.1412.6806.

[275] T. Schnake, O. Eberle, J. Lederer, S. Nakajima, K. T. Schütt, K. R. Müller, G. Montavon. Higher-order explanations of graph neural networks via relevant walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7581–7596, 2022. DOI: 10.1109/TPAMI.2021.3115452.

[276] H. Li, Z. Xu, G. Taylor, C. Studer, T. Goldstein. Visualizing the loss landscape of neural nets. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, Montreal, Canada, pp. 6391–6401, 2018.

[277] M. T. Ribeiro, S. Singh, C. Guestrin. "Why Should I Trust You?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, San Francisco, USA, pp. 1135–1144, 2016. DOI: 10.1145/2939672.2939778.

[278] Z. X. Huang, M. Kosan, S. Medya, S. Ranu, A. Singh. Global counterfactual explainer for graph neural networks. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining*, ACM, Singapore, pp. 141–149, 2023. DOI: 10.1145/3539597.3570376.

[279] Y. Yang, L. Song. Learn to explain efficiently via neural logic inductive learning. In *Proceedings of the 8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2019.

[280] T. L. Wu, H. Y. Ren, P. Li, J. Leskovec. Graph information bottleneck. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 1716, 2020.

[281] Y. X. Wu, X. Wang, A. Zhang, X. N. He, T. S. Chua. Discovering invariant rationales for graph neural networks. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

[282] S. H. Fan, X. Wang, Y. H. Mo, C. Shi, J. Tang. Debiasing graph neural networks via learning disentangled causal substructure. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*,

New Orleans, USA, Article number 1808, 2022.

[283] Y. Q. Chen, Y. G. Zhang, Y. T. Bian, H. Yang, K. L. Ma, B. H. Xie, T. L. Liu, B. Han, J. Cheng. Learning causally invariant representations for out-of-distribution generalization on graphs. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, New Orleans, USA, Article number 1608, 2022.

[284] M. H. Lin, T. Xiao, E. Y. Dai, X. Zhang, S. H. Wang. Certifiably robust graph contrastive learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, New Orleans, USA, 2023.

[285] A. K. Debnath, R. L. L. de Compadre, G. Debnath, A. J. Shusterman, C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, vol. 34, no. 2, pp. 786–797, 1991. DOI: 10.1021/jm00106a046.

[286] Z. Q. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, V. Pande. MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, vol. 9, no. 2, pp. 513–530, 2018. DOI: 10.1039/C7SC02664A.

[287] J. Devlin, M. W. Chang, K. Lee, K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, ACL, Minneapolis, USA, pp. 4171–4186, 2019. DOI: 10.18653/v1/N19-1423.

[288] T. Funke, M. Khosla, A. Anand. Zorro: Valid, sparse, and stable explanations in graph neural networks, [Online], Available: https://arxiv.org/abs/2105.08621, 2021.

[289] E. Dai, S. H. Wang. Learning fair graph neural networks with limited and private sensitive attribute information. *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 7, pp. 7103–7117, 2023. DOI: 10.1109/TKDE.2022.3197554.

[290] B. C. Zhang, Y. S. Dong, C. Chen, Y. D. Zhu, M. N. Luo, J. D. Li. Adversarial attacks on fairness of graph neural networks, [Online], Available: https://arxiv.org/abs/2310.13822, 2023.

[291] H. Xu, X. R. Liu, Y. X. Li, A. Jain, J. L. Tang. To be robust or to be fair: Towards fairness in adversarial training. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 11492–11501, 2021.

[292] J. Adebayo, M. Muelly, I. Liccardi, B. Kim. Debugging tests for model explanations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Vancouver, Canada, Article number 60, 2020.

[293] J. T. Tan, Y. F. Zhang. ExplainableFold: Understanding alphafold prediction with explainable AI. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ACM, Long Beach, USA, pp. 2166–2176, 2023. DOI: 10.1145/3580305.3599337.

[294] X. G. Sun, H. Cheng, J. Li, B. Liu, J. H. Guan. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ACM, Long Beach, USA, pp. 2120–2131, 2023. DOI: 10.1145/3580305.3599256.

[295] Z. K. Chen, H. T. Mao, H. Li, W. Jin, H. Z. Wen, X. C. Wei, S. Q. Wang, D. W. Yin, W. Q. Fan, H. Liu, J. L. Tang. Exploring the potential of large language models (LLMs) in learning on graphs, [Online], Available: https://arxiv.org/abs/2307.03393, 2023.

[296] J. W. Liu, C. Yang, Z. Y. Lu, J. Z. Chen, Y. B. Li, M. M. Zhang, T. Bai, Y. Fang, L. C. Sun, P. S. Yu, C. Shi. Towards graph foundation models: A survey and beyond, [Online], Available: https://arxiv.org/abs/2310.11829, 2023.

Enyan Dai received the B. Sc. degree in mechanical engineering from University of Science and Technology of China, China in and M. Sc. degree in AI from KU Leuven, Belgium in 2016. He is currently working towards the Ph. D. degree at the Pennsylvania State University, USA, under the supervision of Professor Suhang Wang. He has published innovative works in top conference proceedings such as ICLR, WSDM, KDD, CIKM, and ICWSM.

His research interests include data mining, graph neural networks, and trustworthy AI.

E-mail: emd5759@psu.edu

ORCID iD: 0000-0001-9715-0280
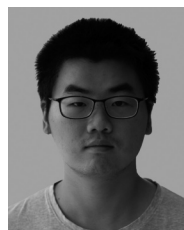
Tianxiang Zhao received the B. Sc. degree in computer science from University of Science and Technology of China (USTC), China in 2017. He is currently working toward the Ph.D. degree in IST at the Pennsylvania State University (PSU), USA, under the supervision of Dr. Suhang Wang and Dr. Xiang Zhang since 2019. He also worked as a research intern at NEC, USA (2021, 2022) and a research intern at Microsoft Research, USA (2023).

His research interests include graph neural networks, weak supervision tasks and knowledge transfer.

E-mail: tkz5084@psu.edu

Huaisheng Zhu received the B. Sc. degree in computer science and engineering from Southern University of Science and Technology of China (Sustech), China in 2020. He is currently working toward the Ph. D. degree in IST at the Pennsylvania State University (PSU), USA, under the supervision of Dr. Vasant Honavar.

His research interests include diffusion models, multimodal generation and molecule generation.

E-mail: hvz5312@psu.edu

Junjie Xu received the B. Sc. degree from Huazhong University of Science and Technology, China in 2021. He is currently a Ph. D. student at The Pennsylvania State University, USA.

His research interests include data mining, graph learning, large language model, and AI for science.

E-mail: junjiexu@psu.edu

Zhimeng Guo received the B. Sc. degree in computer science from the University of Electronic Science and Technology of China (UESTC), China in 2021. He is currently working toward the Ph. D. degree in IST at the Pennsylvania State University (PSU), USA, under the supervision of Dr. Minhao Cheng.

His research interests include robustness and alignment of foundation models.

E-mail: zzg5107@psu.edu

Hui Liu received the Ph. D. degree in electrical engineering from southern Methodist University, USA in 2015. She is an assistant professor in the Computer Science and Engineering Department at Michigan State University, USA.

Her research interests include trustworthy AI, designing data mining algorithms for wireless communication of smart devices, and applying machine learning and data mining in wireless communications.
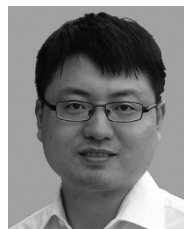
E-mail: liuhui7@msu.edu

Jiliang Tang received the Ph. D. degree in computer science from Arizona State University, USA in 2015. He is an associate professor in the Computer Science and Engineering Department at Michigan State University, USA. Before that, he was a research scientist in Yahoo Research, USA. He was the recipient of the 2021 ACSIC Rising Star Award, 2021 IEEE Big Data Security Junior Research Award, 2020 ACM SIGKDD Rising Star Award, 2020 Distinguished Withrow Research Award, 2019 NSF Career Award and 7 best paper (or runner up) awards including KDD2015 and WSDM2018. His dissertation won the 2015 KDD Best Dissertation runner-up and Dean's Dissertation Award. He has served as the editors and the organizers in prestigious journals (e.g., TKDD) and conferences (e.g., KDD, WSDM, and SDM). He has filed more than 10 US patents and has published his research in highly ranked journals and top conference proceedings, which received more than 17 000 citations with h-index 62 and extensive media coverage.

His research interests include data mining, machine learning and their applications on social, web, and education domains.

E-mail: tangjili@msu.edu

Suhang Wang received the B. Sc. degree in electrical and computer engineering from Shanghai Jiao Tong University, China in 2012, the M. Sc. degree in electrical engineering from University of Michigan–Ann Arbor, USA in 2013, the Ph. D. degree in computer science from Arizona State University, USA in 2018. He is an assistant professor of the College of Information Sciences and technology, the Pennsylvania State University, USA. He is an Associate Editor for several journals and serves as regular journal reviewers and numerous conference program committees. He has published innovative works in highly ranked journals and top conference proceedings such as IEEE TKDE, ACM TIST, KDD, WWW, AAAI, IJCAI, CIKM, SDM, WSDM, ICDM and CVPR, which have received extensive coverage in the media.

His research interests include graph mining, data mining and machine learning.

E-mail: szw494@psu.edu (Corresponding author)

ORCID iD: 0000-0003-3448-4878