Trustworthy Graph Neural Networks: Aspects, Methods, and Trends

This article surveys efforts from the perspective of “trustworthy graph neural networks,” defined as those that incorporate core aspects such as robustness, explainability, privacy, fairness, accountability, and well-being.

By He Zhang✉, Bang Wu, Xingliang Yuan✉, Senior Member IEEE, Shirui Pan✉, Senior Member IEEE, Hanghang Tong, Fellow IEEE, and Jian Pei✉, Fellow IEEE

Abstract
Graph neural networks (GNNs) have emerged as a series of competent graph learning methods for diverse real-world scenarios, ranging from daily applications such as recommendation systems and question answering to cutting-edge technologies such as drug discovery in life sciences and n-body simulation in astrophysics. However, task performance is not the only requirement for GNNs. Performance-oriented GNNs have exhibited potential adverse effects, such as vulnerability to adversarial attacks, unexplainable discrimination against disadvantaged groups, or excessive resource consumption in edge computing environments. To avoid these unintentional harms, it is necessary to build competent GNNs characterized by trustworthiness.

To this end, we propose a comprehensive roadmap to build trustworthy GNNs from the view of the various computing technologies involved. In this survey, we introduce basic concepts and comprehensively summarize existing efforts for trustworthy GNNs from six aspects, including robustness, explainability, privacy, fairness, accountability, and environmental well-being. In addition, we highlight the intricate cross-aspect relations between the above six aspects of trustworthy GNNs. Finally, we present a thorough overview of trending directions for facilitating the research and industrialization of trustworthy GNNs.

Keywords
Accountability; environmental well-being; explainability; fairness; graph neural networks (GNNs); privacy; robustness; trustworthy machine learning.

I. Introduction
Graphs are a ubiquitous data structure used to represent data from a broad spectrum of real-world applications. They have demonstrated a remarkable capacity to represent both objects and the diverse interactions between them. For example, a single graph depicting friendship between users on Facebook can have 2.9 billion nodes and more than 490 billion edges [1]. Recently, graph neural networks (GNNs) [2], [3] have emerged as a series of powerful machine learning methods for exploring graph data and have made impressive advancements in areas ranging from daily applications to technological frontiers. In consumer applications, GNNs enrich personalized search and recommendations for customers on e-commerce (e.g., Alibaba [4]) and social media (e.g., Pinterest [5]) platforms by considering user-user/item interactions. In the advanced sciences, researchers use graphs to represent complex systems (e.g., physics simulations [6]) and employ GNNs to explore the objective laws that govern
celestial motion [7]. GNNs are also conducive to improving well-being in our society, with applications ranging from fake news detection [8] to discovering drugs that treat COVID-19 [9].

Many architectures and methods have been proposed to improve the performance of GNNs from different perspectives [16], [17], including enhancing their expressive power [18], overcoming oversmoothing issues [19], increasing architecture depths [20], and utilizing self-supervised signals [21], to name only a few. In critical and sensitive domains, however, competent performance (i.e., high accuracy) is not the only objective. For example, in the context of GNN-based anomaly detection systems, it is crucial for such systems to be robust against adversarial attacks [22]; in GNN-based credit scoring systems, it can be unethical and raise societal concerns if such systems decline a loan application based on potential biases defined by attributes such as gender, race, and age [23]; and in GNN-based drug discovery systems, it is desirable for the process of identifying screened drug candidates to be explainable and comprehensible by humans [24]. Driven by these diverse requirements, people increasingly expect GNNs to be reliable, responsible, ethical, and socially beneficial enough to be trusted. In this survey, we aim to provide a timely survey to summarize these efforts from the perspective of “trustworthy GNNs.”

### A. Trustworthiness

The core of a trustworthy system lies in its trustworthiness. As defined in the Oxford Dictionary, “trustworthy” describes an object or a person “that you can rely on to be good, honest, sincere, and so on” [25]. Synonyms of trustworthy include trustable, reliable, dependable, faithful, honorable, creditworthy, responsible, and so on. The trustworthiness of a system essentially builds on the “trust” that makes the system “worthy” of being relied on. There are a number of fields in which “trust” is defined and studied, including philosophy, psychology, sociology, economics, technology, and organizational management [10], [12], [13]. It is widely recognized that trust relates to people’s perceived ability to rely on others (e.g., persons or systems); trust, in fact, can be considered the foundation of social order [26].

Trust can also be defined as “the relationship between a trustor and a trustee: the trustor trusts the trustee” [12]. In the context of trustworthy GNNs, we define the trustee as the GNN models or systems and the trustor as the model owners, users, regulators, or even our society as a whole. A GNN model (system) is trustworthy if it is developed with the key aspects of trustworthiness in mind, as we will discuss later.

### B. From Trustworthy AI to Trustworthy GNNs

In recent years, a global consensus has developed as to the importance of building trustworthy AI. Continuing efforts have been made to promote the development of ethical, beneficial, responsible, and trustworthy AI [10]. These efforts are presented from views ranging from guidelines and mechanisms to technology (as shown in Table 1). In Table 1, we present the views and contents of existing efforts related to trustworthy AI.

The research focusing on guidelines proposes a set of principles and requirements for building responsible AI [14], [15]. For example, the safety and controllability principle [14] indicates that “the transparency, interpretability, reliability, and controllability of AI systems should be improved continuously to make the systems more traceable, trustworthy, and easier to audit and monitor.” From the view of developing mechanisms, Brundage et al. [13] made recommendations for supporting verifiable claims. For example, no formal process or incentive exists for individuals who are not affiliated with a particular AI developer to report safety-and-bias-related issues in AI systems; therefore, it is relatively rare for AI systems to be broadly scrutinized on these issues. To this end, the authors recommended an institutional mechanism in

### Table 1 View and Contents of Typical Concepts Related to Trustworthy AI

| View     | REF | Robustness | Explainability | Privacy | Fairness | Accountability | Well-being | Others                  |
|----------|-----|------------|----------------|---------|----------|----------------|------------|-------------------------|
| Technology | [10] | Safety, Robustness, Adversarial Robustness | Interpretability, Transparency | Privacy, Consent, Privacy Protection | Non-discrimination, Fairness | Accountability, Auditability, Reproducibility, Transparency | Environmental Well-being, Value Alignment, Professional Codes & Ethics Guidelines, Lived Experience, Disinformation & Filter Bubbles, Social Good | Generalisation, Distribution Shift |
| Mechanism | [13] | Red Teaming Exercises | Interpretability | PPML Standardization | – | Bias Safety Bounties, Sharing of AI Incidents, Audit Trail | High-Precision Compute Measurement | – |
| Guideline | [14] | Harmony & Human-Friendly, Fairness & Justice, Inclusion & Sharing, Respect for Privacy, Safety & Controllability, Shared Responsibility, Open & Collaboration, Agile Governance | | | | | |
| | [15] | Well-Being, Respect for Autonomy, Privacy & Intimacy, Solidarity, Democratic Participation, Equity, Diversity Inclusion, Prudence, Responsibility & Sustainable Development | | | | | |
which developers of trustworthy AI systems should pilot bias and safety bounties. From the technology perspective, a few studies have summarized advancements in trustworthiness derived from principles outlined by existing guidelines [10], [11]. For example, the privacy of trustworthy AI can be enhanced by confidential computing [27], federated learning (FL) [28], and differential privacy (DP) technologies [10], [29].

Although the existing literature explores the space of trustworthy AI from different views (i.e., “Technology,” “Mechanism,” and “Guideline” in Table 1), several key aspects from the technology view receive the most recognition, namely, robustness, explainability, privacy, fairness, accountability, and well-being. As a vital instantiation of trustworthy AI in the context of GNNs, we transfer these trustworthiness aspects from general AI to GNN systems and present an open framework (as shown in Fig. 1) on trustworthy GNNs. In practice, the characteristics of graph data differentiate the research with a specific focus on trustworthy GNNs from that exploring trustworthy AI. In Section I-C, we will introduce these core aspects in

---

**Fig. 1.** Open framework of trustworthy GNNs and overview of methodology categorization. Trustworthy GNNs are competent GNNs that incorporate aspects of trustworthiness, including robustness, explainability, privacy, fairness, accountability, well-being, and other trust-oriented aspects in the context of GNNs. For each aspect of trustworthy GNNs, we comprehensively summarize typical methods and provide methodology categorization, as will be shown in Sections III–VIII.
In this survey, we define **trustworthy GNNs** as competitive GNNs that incorporate core aspects of trustworthiness, including robustness, explainability, privacy, fairness, accountability, well-being, and other trust-oriented characteristics in the context of GNNs. We believe that incorporating these aspects when designing GNNs will significantly improve their trustworthiness and promote their industrialization and broad applications.

### C. Aspects of Trustworthy GNNs

From the view of deep learning, GNNs provide a generalized way to explore graph data. Unlike common image data, graph data are derived from non-Euclidean space and have certain unique characteristics (e.g., discreteness, irregularity, and non-independent identically distributed (i.i.d.) nodes), which distinguish trustworthiness practices in GNNs from those in general AI. To better illustrate our trustworthy GNN framework, we here briefly explain each core aspect of trustworthiness, describe the metrics used for evaluation, and pinpoint the differences between research into trustworthy GNNs and general trustworthy AI (also summarized in Table 2).

1) **Robustness**: Robustness refers to the ability of GNNs to remain stable under perturbations, especially those that are created by attackers. For example, a credit rating GNN in a financial system should remain secure despite a perturbed transaction graph in which attackers attempt to forge connections between users with high credit scores.

   a) **Metrics**: Generally, the robustness of GNNs can be measured with reference to the model accuracy [55], attack success rate of certain attack algorithms [30], misclassification rate [56], structural similarity score [57], and attack budget [58].

   b) **Research differences**: Since graph structure is highly discrete, the notion of perturbations on graph samples (e.g., adding/deleting edges) is fundamentally different from those on ordinary samples in the Euclidean space (e.g., images) [46].

2) **Explainability**\(^2\): Similar to general deep learning models, most GNNs are regarded as black-box models due to the absence of explainability. Explainability enables the predictions of GNNs to be traceable, which, in turn, enables humans to understand the behaviors of GNNs and discover knowledge. For instance, GNN explanations (e.g., subgraphs) can reveal which components in molecule graphs support the final biochemical functionality predictions of GNNs [59].

   a) **Metrics**: Explanation accuracy [31], explanation faithfulness [32], explanation stability [60], and explanation sparsity [61] are commonly used metrics for evaluating GNN explanations. In specific applications, some metrics based on domain knowledge (e.g., correlated separability [62] in computational pathology) are also used to measure explanations.

   b) **Research differences**: Unlike explanations for general AI (e.g., saliency maps and subimages), GNN explanations (e.g., nodes, edges, and subgraphs) inherit the irregularity and discreteness of graph data, which go beyond the data characteristics in general AI (e.g., regularity of image data). Thus, the generation of GNN explanations requires specially designed methods [31].

3) **Privacy**: Privacy indicates how private data within GNNs can be protected to prevent it from being leaked. Specifically, the privacy of graph data and model parameters, which are regarded as confidential information belonging to their owners, should be guaranteed, and any exposure to unauthorized parties should be prevented [47, 63].

   a) **Metrics**: Generally, a private GNN requires that no leakage of private data (e.g., nodes, edges, the graphs themselves, GNN model parameters, and hyperparameters) is fundamentally different from those on ordinary samples in the Euclidean space (e.g., images) [46].

   **Table 2** Typical Metrics for Aspects in Trustworthy GNNs and Typical Research Differences Between Trustworthy GNNs and AI

| Aspects        | Robustness     | Explainability | Privacy        | Fairness       | Accountability | Environmental Well-being |
|----------------|----------------|----------------|----------------|----------------|----------------|--------------------------|
| Typical Metrics | ASR\(^*\) Accuracy [30] | ASR\(^*\) Data Leakage [33] | Group/Individual Fairness [34,35] | Standard Evaluations [36] | Inference Time [37], Nodes-Per-Joule [38] |

\(^*\) ASR indicates the attack success rate of adversarial/privacy attacks on robustness/privacy. IID stands for independent and identically distributed.

\(^\) We will expand on the difference between interpretability and explainability in Section IV-A.
for GNN training) occurs in its systems [64]. Privacy can also be measured based on the ability of GNNs to defend against privacy attacks and reduce their attack success rates [65].

b) Research differences: Graph data, as an important component of GNNs, impose additional privacy requirements. In addition to protecting the sensitive information of the entities (i.e., nodes), the relationships between them (i.e., edges) also need to be protected [66].

4) Fairness: Discrepancies in predictions made with regard to marginalized groups (e.g., female or minorities) indicate that the behavior of some GNNs is unfair. Recent studies have shown that GNNs can discriminate against samples based on protected and sensitive attributes such as gender and nationality (e.g., discrimination in credit risk prediction [67]). Fairness requires GNNs to accommodate differences between people and provide the same quality of service to users from diverse backgrounds.

a) Metrics: Currently, group fairness (e.g., demographic parity [68] that requires no dependence on predictions on membership in a sensitive group), individual fairness (e.g., similarity-based fairness [35] that expects similar individuals are treated similarly), and counterfactual fairness [67], [69] are commonly used metrics for evaluating the fairness of GNNs.

b) Research differences: Unlike the data distribution in general AI fairness, the existence of edges between nodes breaks the assumption that samples are i.i.d. This implies that specific fairness definitions (e.g., involving a similarity function defined on edges) are necessary when studying fairness in GNNs for node-focused computational tasks (see Section II); examples include node classification [35] and link prediction [34].

5) Accountability: Accountability is another aspect of trustworthy GNNs that consider the ability to determine whether a system is performing in accordance with procedural and substantive standards [70]. Accountable GNNs contain a set of specific assessments and metrics to justify and identify the responsibilities associated with individual roles or development steps in GNNs.

a) Metrics: The accountability of GNNs can be measured by whether a standard evaluation process exists [36] and how comprehensive the specifications are established for each step or role of the GNN life cycle [44]. For instance, a more comprehensive accountable GNN system would be assessed through its detailed development life cycles as distinct stages (e.g., accounting for the software development life cycle (SDLC), including planning, analysis, design, implementation, testing/integration, and maintenance) [44], [71].

b) Research differences: Due to the unique characteristics of graph data, the evaluation standards for GNNs differ from those of other deep learning methods. For example, graph datasets with different graph-based metrics (e.g., node degrees) are used to assess how GNN architectures perform differently on various datasets [36]. Therefore, researchers need to design new procedures and techniques specifically for building an accountable GNN.

6) Environmental Well-Being: Well-being evaluates if GNNs are aligned to people’s expectations regarding social good.³ There is currently an urgent need to deploy GNNs on edge devices for real-time inference systems with limited processing power and memory resources (e.g., point cloud segmentation in autonomous vehicles [75]). As a representative subaspect of well-being, environmental well-being focuses on measuring the efficiency of GNNs.

a) Metrics: Generally, resource-/efficiency-related metrics, such as inference time [37], memory footprint [76], or nodes-per-joule [38], are employed to evaluate efforts on environmental well-being of GNNs.

b) Research differences: The efficiency bottlenecks of general AI are caused by the scale [10] and energy-intensive architecture [45] of models. Due to the powerful presentation capacity and characteristics of graphs, the graph scale and data irregularity are the primary sources of the efficiency bottleneck in GNNs, whose operations rely on input graph structures [73].

³Well-being is a generalized term, which encompasses both environmental well-being and various other forms of well-being (ranging from generating long-term value for shareholders to ending poverty [12]). In contrast to other forms of well-being, environmental well-being has mainly been explored from the technology perspective. Thus, in this survey, discussions of well-being will focus on introducing existing advancements in environmental well-being.
D. Related Surveys

Related surveys include reviews on GNNs and trustworthy AI, along with reviews on a single or multiple aspects of trustworthy GNNs (as shown in Table 3). Compared with those surveys, our survey elaborates on existing advancements in the above six aspects of trustworthy GNNs and explores the complex relations between them.

**Surveys on GNNs** mostly focus on performance-oriented methods [2], [3], [72]. Wu et al. [2] review several different GNN architectures, including recurrent GNNs, convolutional GNNs, graph autoencoders, and spatial–temporal GNNs. Ruiz et al. [3] review GNNs from the perspectives of graph perceptrons, multiple-layer networks, and multiple-feature networks. Zhang et al. [72] categorize deep learning methods on graphs into graph recurrent neural networks, graph convolutional networks, graph autoencoders, graph reinforcement learning (RL), and graph adversarial methods by distinguishing the basic assumptions/aims that underpin these methods.

**Surveys on trustworthy AI** review research into general AI systems [10]. Due to the unique characteristics of graph data and GNNs (e.g., the interdependence between nodes, and intertwined computation between dense and sparse operations during GNN inferences [73]), these surveys do not appear to be able to thoroughly guide building trustworthy GNN systems. For example, since graph structure is highly discrete, the notion of perturbations on graph samples is intrinsically different from that of perturbations on ordinary samples in the Euclidean space [46].

**Surveys on a single aspect/multiple aspects of trustworthy GNNs** have also emerged recently [30], [31], [73]. Jin et al. [30] review existing adversarial attack methods and corresponding defense strategies for GNNs. Yuan et al. [31] provide an overview of posthoc explainer methods for GNNs by discussing the methodologies employed. Abadal et al. [73] summarize current software (SW) frameworks and hardware (HW) accelerators for GNNs. Note that these authors do not comprehensively consider how to build trustworthy GNNs. Some of them focus on subtopics of a particular aspect of trustworthy GNNs. For example, the survey [31] of explanation methods for GNNs mainly reviews posthoc explainers.

Concurrent to our survey, Dai et al. [74] present an insightful review on trustworthy GNNs from four aspects, namely, privacy, robustness, fairness, and explainability. Our survey differs from this work in three key ways.

1) **Consolidated and fine-grained taxonomy:** For each trustworthiness aspect, we provide a specific definition of the aspect, metrics for evaluation, and a fine-grained taxonomy. For example, defense methods for robustness are categorized into pretraining, during training, and posttraining methods. Our comparison and discussion of defense methods enable practitioners to build robust GNNs according to their demands during different phases.

2) **Cross-aspect relations:** Our survey pinpoints the complex relations between aspects of trustworthy GNNs. In particular, our survey provides insights on: a) how the methods from one aspect of trustworthiness are adapted to address objectives in other aspects and b) why advancing one aspect of trustworthy GNNs can promote or inhibit other aspects. These relations are vital to a comprehensive understanding and construction of trustworthiness.

3) **Open framework:** Our survey provides an open framework (see Fig. 1) that includes six key aspects that have attracted increasing research attention in the field. This framework is extensible and, thus, capable of incorporating any aspect of trustworthiness that is yet to be considered but may be necessary for the future.

Table 3 presents a comparison between our survey and representative surveys on GNNs, trustworthy AI, and studies on one or multiple aspects of trustworthy GNNs. Taking a different approach from the above surveys, our survey thoroughly reviews current trust-oriented methods for constructing trustworthy GNN systems. We aim to provide a methodical roadmap for both
researchers and industry practitioners working in the GNN community.

E. Contributions and Organization

The contributions of our survey can be summarized as follows.

1) Open framework for trustworthy GNNs: We propose an open framework containing trust-oriented characteristics for use in accurately characterizing trustworthy GNNs. To illustrate the core aspects of this framework, we present their evaluation metrics along with the differences between research into trustworthy GNNs and that focusing on general AI.

2) Comprehensive methodology categorization: For each aspect of trustworthy GNNs, we comprehensively summarize the basic concepts and typical methods and discuss future research directions.

3) Insights for cross-aspect relations: We elaborate on the complex relations between the aspects in our open framework in terms of methodologies and achievements, which are vital to consider if fully trustworthy GNNs are to be achieved.

4) Outlook on trending directions: By considering trustworthy GNNs as a whole and summarizing the common limitations of current advancements, we point out several promising avenues of exploration.

The remainder of this survey is organized as follows. We first introduce some key concepts related to GNNs in Section II. Working from the definition of trustworthy GNNs, we summarize current methods for improving the trustworthiness of GNNs from different aspects in Sections III–VIII and then discuss the complex cross-aspect relations in Section IX. Finally, we conclude this survey in Section X and point out trending directions of research and industrialization of trustworthy GNNs.

II. CONCEPTS

A. Graphs

A graph can be denoted as \( G = (V, E) \), where \( V = \{v_1, \ldots, v_{|V|}\} \) is a set of nodes and \( E \) is a set of edges. \( E \) describes the structural information of \( G \), which can also be expressed as the adjacency matrix \( A \in \{0, 1\}^{|V| \times |V|} \). \( A_{i,j} = 1 \) if \( e_{ij} = (v_i, v_j) \in E \); otherwise, \( A_{i,j} = 0 \). The edge feature associated with edge \( e_{uv} \) is \( e_{uv} \). The node features are expressed as a matrix \( X \in \mathbb{R}^{|V| \times d} \), whose \( i \)th row indicates the feature values of \( v_i \), and \( d \) presents the dimensionality of features. Thus, a graph can also be expressed as \( G = (A, X) \).

A subgraph \( G_s = (V_s, E_s) \) of \( G \) is composed of a node subset \( V_s \subseteq V \) and an edge subset \( E_s \subseteq E \), where \( V_s \) contains all nodes involved in \( E_s \). Correspondingly, when features are associated with nodes, a subgraph can also be expressed as \( G_s = \{A_s, X_s\} \).

B. Graph Neural Networks

GNNs are a series of neural network architectures designed to explore graphs by learning graph embedding [83], [84], [85]. We here introduce typical message-passing-based GNNs, in which the node embedding is iteratively updated as follows [83]:

\[
\begin{align*}
M_t & = \sum_{u \in \mathcal{N}(v)} M_t^u \left( h_u^{t-1}, h_v^{t-1}, e_{uv} \right) \\
h_v^t & = U_t \left( h_v^{t-1}, m_v^t \right)
\end{align*}
\]

where \( h_v^t \) indicates the embedding of node \( v \) at layer \( t \in \{1, \ldots, T\} \) and \( \mathcal{N}(v) \) denotes the set of neighbors of \( v \) in graph \( G \). \( M_t(\cdot, \cdot, \cdot) \) and \( U_t(\cdot, \cdot) \) are the message function and the embedding updating function at layer \( t \), respectively. By means of the above operations, message-passing-based GNNs can provide graph embedding for various tasks.

C. Computational Tasks

Tasks on graphs can be categorized into node-focused tasks and graph-focused tasks [54]. Node-focused tasks mainly include node classification, node ranking, link prediction, and community detection [86]. Graph-focused tasks mainly include graph classification, graph matching, and graph generation [54]. Two representative tasks are discussed in more detail in the following.

1) Node Classification: Given a graph \( G = (V, E) \), \( V_1 \subseteq V \) denotes the set of labeled nodes, and the label associated with \( v_i \in V_1 \) is \( y_i \). \( V_u = V \setminus V_1 \) is the set of other unlabelled nodes. The goal of node classification is to learn a GNN model \( f_\theta \) from \( G \) and node labels and then utilize \( f_\theta \) to predict labels for the nodes in \( V_u \).

2) Graph Classification: Given a graph dataset \( D = \{(G_i, y_i)\} \), in which \( y_i \) is the label of a graph \( G_i \), the goal of graph classification is to learn a GNN model \( f_\theta \) from \( D \) and then use \( f_\theta \) to predict labels for unseen graphs.

More details about other GNN architectures (i.e., architectures beyond the message-passing mechanism) and tasks (e.g., link prediction and graph generation) can be found in several books [54], [87] that provide an introduction to GNNs.

III. ROBUSTNESS OF GNNs

Robustness is one of the most critical aspects of trustworthy GNN development. In general terms, robustness refers to the ability of systems to perform persistently under a variety of conditions. In the context of GNNs, a robust GNN can sustain model accuracy under perturbations such as malicious graph structure modifications made by adding or deleting edges. Recent studies...
A. Adversarial Attacks

1) Threat Models and Attack Categories: Threat modeling represents how an attacker might use adversarial perturbations to disrupt the performance of a GNN model. There are several aspects in the threat model of GNN adversarial attacks, including when an attack takes place, what information an attacker can obtain, what adversarial actions can be taken, and how the attacker might attempt to harm the performance of the targeted GNN model. In this section, we will introduce different types of attacks based on their threat models. Fig. 2 illustrates the overall processes of several common attacks. The four dashed-line boxes (from left to right) illustrate how evasion attacks, no attack, poisoning attacks, and backdoor attacks are applied to GNNs during their training (the top blue box) and/or inference (the middle yellow box) phases. The bottom red box presents different attack results. Next, we will introduce these attacks in GNNs by presenting their attack stages, attacker knowledge, perturbation objects, and attack goals.

a) Attack stages: Attacks can occur either during the development phase (also known as training) or the deployment phase (also known as inference). Generally, these attacks can be divided into two categories: poisoning attacks and evasion attacks.

i) Poisoning attacks: Poisoning attacks target the training phase of GNN model development. Attackers attempt to alter the training graphs of a target GNN, leading to the production of a poisoned model with impaired performance [94], [95], [96].

ii) Evasion attacks: Attacks that target the inference phase are known as evasion attacks. Attackers aim to perturb the graphs during the inference period so that the perturbed graph can cause the clean GNN models to engage in incorrect behaviors (e.g., misclassification) [55], [97], [98]. Different from poisoning attacks, attackers in evasion attacks cannot affect the parameters of the target GNN models. Therefore, it is assumed that the target GNN models have been well-trained on the clean graph and are considered to be fixed during this type of attack.

b) Attacker’s background knowledge: In practical scenarios, an attacker may be able to gather different types of knowledge. This knowledge typically consists of the target GNN model (e.g., model parameters and architectures) and the training or testing data. Based on the type and degree of knowledge obtained, attacks can be classified into three categories: white-box attacks, black-box attacks, and gray-box attacks.

i) White-box attacks: In a white-box attack, attackers have access to the entire target GNN, including its architecture, parameters, and gradient [89], [100]. This information can be used to construct adversarial graphs; for example, attackers may seek the desired perturbations by calculating the gradient corresponding to prediction errors (see details in Section III-A2). Note that white-box attacks require the strongest assumption with regard to knowledge of the target system, which may be not available in real-world applications.

ii) Black-box attacks: By contrast, a black-box attack means that attackers do not know anything about the target GNN model and can only send queries to GNNs [55], [96]. A common approach [46], [55] to conducting a black-box attack involves building a surrogate model. Specifically, attackers manage to gather information from the target GNN model (e.g., exploring the mapping between designed input queries and their responses) to construct a surrogate model. Then, with full access to the surrogate model, attackers can utilize white-box attack strategies to generate adversarial graph samples. It is worth noting that compared with white-box attacks, black-box attacks are more dangerous since they are more applicable to real-world scenarios.
Fig. 2. Process of different adversarial attacks against GNNs. Training and inference without adversarial perturbations can produce a clean GNN model and correct prediction results (as shown in the second column dotted box). Meanwhile, different perturbations to the clean graph are added during either a training or inference phases to produce evasion attacks (the first column dotted box), poisoning attacks (the third column dotted box), and backdoor attacks (the fourth column dotted box), which lead to different kinds of misclassifications (“Incorrect Results” box at the bottom).

c) Perturbation operations: An attacker’s capability is also defined in the threat model. Most existing attacks propose to inject perturbations into graph data. Specifically, there exist multiple types of perturbations, including the following.

i) Perturbing graph structure: Most existing studies propose to implement their attacks by modifying edges (i.e., adding/deleting/rewiring edges) [55], [89], [101]. In practice, attackers often constrain their perturbations within a given budget (e.g., making modifications to only a limited number of edges), which makes them more difficult to detect.

ii) Perturbing node attributes: In addition, perturbations can also be added to node attributes [102]. Notably, empirical studies have shown this approach to be less effective than perturbations on edges [46], [103]; consequently, it is often used in combination with other types of perturbations [46], [94].

iii) Injecting nodes: Another practical perturbation is to inject malicious nodes into graph data [88], [96], [100], [104]. These malicious nodes can be linked to benign nodes, leading to misclassifications. This type of perturbation is attractive when attackers cannot modify existing edges. For example, it is difficult for attackers to break a credit ranking system and tamper with existing connections but easier for them to register a new account and perform malicious actions such as adding new connections with other accounts.

d) Attack goals: Attack goals describe how attackers expect the performance of GNN models to be reduced. In general, they can be described from two perspectives.
i) **Error specificity:** Attackers may expect different types of errors to result from their adversarial attacks. For example, in a node/graph classification task, an error-specific attacker expects the target GNN to misclassify nodes/graphs under a specific label, while error-nonspecific attackers only expect misclassifications more generally without specific target labels.

ii) **Attack specificity:** Attackers may have different attack targets. Specifically, in node-level GNNs, some attackers [46] expect only a small set of nodes to be misclassified, while others aim to degrade the overall performance of the target GNNs [94]. It should be noted that existing studies [46], [94] of GNNs also refer to the former as targeted attacks and the latter as nontargeted attacks. This convention differs from that used for attacks in nongraph fields (e.g., images), where misclassifying inputs under a specific label is known as a targeted attack, while misclassifying inputs to any incorrect label is known as a nontargeted attack.

2) **Attack Methods:** In this section, we will describe several representative attack methods.

a) **Attack formulation:** To perform the above attacks, attackers formalize the construction of perturbations as one of several optimization problems. Specifically, attackers aim to find a perturbed graph that can reduce the performance (e.g., overall accuracy on the testing set) of GNN models. The objective functions and constraints of the formalized optimization problem can be determined with reference to the aspects introduced in Section III-A1. Here, we provide an example of the problem formalization from Nettack [46]: given a target graph $G = \{A, X\}$ and a target node $v_i$ with ground-truth label $y_i$, an attacker attempts to construct a perturbed graph $\tilde{G}$ with limited perturbations such that the GNN model $f_{\theta^*}$ trained on $G$ undergoes the maximum performance drop for the node classification task

$$\max_{G = \{\tilde{A}, \tilde{X}\}} I\left(f_{\theta^*}(\tilde{G}, v_i) \neq y_i\right)$$

s.t. $D(\tilde{G}, G) < \epsilon$

$$\theta^* = \arg\min_{\theta} L\left(\theta; \tilde{G}\right)$$

where $I(\cdot)$ is a binary indicator function, $D(\cdot, \cdot)$ is a perturbation measurement function to assess the distance between the clean graph $G$ and the perturbed graph $\tilde{G}$, and $\epsilon$ is a perturbation budget.

Considering that Nettack is a poisoning attack (it can also be adapted as an evasion attack), it directly perturbs the training graph from $G$ to $\tilde{G}$, so as to generate the target GNN $f_{\theta^*}(\cdot, \cdot)$. In addition, the attacker simultaneously perturbs the graph structure and node attribute values so that the generated perturbed graph $\tilde{G}$ includes both of these perturbations $\{\tilde{A}, \tilde{X}\}$. Finally, it aims to cause misclassification of a specific node; thus, the objective function of the attack is designed to measure the error between the prediction of $f_{\theta^*}(\tilde{G}, v_i)$ and its ground-truth label $y_i$.

b) **Optimization solving:** As a next step, attack methods are proposed via solving optimization functions. Two main strategies are employed to achieve this, namely, gradient-based and nongradient-based methods.

i) **Gradient-based methods:** Since white-box attacks are able to utilize information about gradients, attackers can employ gradient-based methods [89], [100]. Specifically, by regarding the desired perturbed graph as a hyperparameter, attackers can calculate the partial derivative of the loss (e.g., the loss between the predictions and ground-truth labels) with respect to input graphs. These optimizations can be performed using gradients that are very similar to the training gradients [105]. Nevertheless, the difficulty lies in the discreteness of graph data. There are several approaches to solving such a problem [55], [106], [107]. One common method is to greedily and iteratively perturb the graph based on the gradient [55], [108]. Alternatively, the attacker can use techniques such as integrating gradients, among others, to convert the discrete data to continuous data. For example, the integrating gradients of a loss function with respect to an edge perturbation on the input graph are represented as follows [107], as shown in the equation at the bottom of the next page, where $A$ is the adjacency matrix, $A_{ij}$ specifies the perturbed edges, and $m$ is the step number for computing integrating gradients. In this case, integrating gradients can be regarded as the importance scores for edge perturbations, where a higher value indicates that the perturbation will result in better attack effectiveness. After calculating all the gradients, the attacker can greedily find the perturbation that produces the final adversarial graph.

ii) **Nongradient-based methods:** In black-box attack scenarios, however, the gradients are unknown to attackers. There are several approaches [55], [96], [109] designed for solving the optimization problem without using gradients, such as RL [96] and genetic algorithms [55]. Specifically, attackers utilizing RL algorithms can define graph perturbations as executing actions and then design rewards based on their attack goals [55]. The attack procedure can be modeled as a finite horizon Markov decision process (FHMDDP). a) **Action ($a$):** A single action at time step $t$ is denoted as $a_t$, which represents adding or deleting an edge. b) **State ($s$):** The state is represented as a tuple $(G^t, u_i)$, where $G^t$ is the perturbed graph at time step $t$ and $u_i$ is the attacker’s target node. c) **Reward ($r$):** The attacker receives no reward in the intermediate steps, i.e., $\forall t = 1, 2, \ldots, m - 1, r(s_t, a_t) = 0$. 

Meanwhile, the attacker receives a nonzero reward at the end of FHMDP

\[ r(s_m, a_m) = \begin{cases} 1, & \text{if } f_\theta(G^m_m, u_i) \neq y_i, \\ -1, & \text{if } f_\theta(G^m_m, u_i) = y_i. \end{cases} \]

d) **Terminal:** The process terminates after modifying \( m \) edges. Furthermore, genetic algorithms can also be used to select the population (i.e., a perturbed graph) based on a fitness function that iteratively evaluates attack effectiveness [55].

**c) Comparison:** In addition to attack stages, perturbation objects, and attack goals that affect the formulation of the attack, optimization solving methods are determined by the degree of attacker knowledge. White-box attacks, for example, use gradient information to construct their perturbations, while black-box attacks make use of techniques other than gradients, such as greedy search or RL. Gradients offer the advantage of being straightforward and easy to implement. A number of works use this approach as a baseline since it illustrates the vulnerability of victim models in the worst-case scenario [110].

In practice, however, an attacker may not have access to the gradients of the victim models. Thus, rather than utilizing the gradients of the victim models directly, some black-box attacks [88] construct surrogate models and use the gradients obtained therefrom. Another study in black-box settings [111] uses adversarial graph search and query responses of the victim models to calculate the sign gradients. These approaches may require additional effort to determine the gradients (e.g., the time cost for the topology attack using a surrogate model can be six to ten times that of a simple gradient-based approach [112]). Meanwhile, nongradient methods are also widely employed in many studies [55], [109]. Most of these methods do not require knowledge of gradients, which makes them more practical when dealing with situations in which an attacker has limited knowledge.

**B. Defenses**

The methods employed to defend against the above attacks can be classified based on the targeted phases of GNN systems, i.e., defenses occurring before, during, and after training.

1) **Defenses Before GNN Training:** Since most attacks tend to perturb the graph data, it is natural to consider preprocessing the graph data before training.

   a) **Training graph preprocessing:** Graph preprocessing aims to distill current graph data by removing potential adversarial perturbations. As studies [107], [113] have shown, many perturbations applied by attackers have specific characteristics. By referring to these characteristics, defenders can identify suspicious components and remove them before GNN training begins. Defenders can also compute the low-rank approximation of the adjacency and feature matrices and then discard any high-rank perturbations [113].

   b) **Clean graph learning:** These types of defenses propose to directly regenerate a clean graph rather than deleting a small number of suspicious edges. The intuition supporting the use of graph learning techniques in these scenarios is that adversarial perturbations normally conflict with the basic graph property [114]. Thus, graphs reconstructed using these graph structure learning methods can erode the negative effects of the adversarial perturbations [107]. In addition, graph sanitization has also been employed by learning a “better” training graph from a perturbed graph [115].

2) **Defenses During GNN Training:** Alternatively, some defense methods aim to improve the GNN robustness during the training process.

   a) **GNN architecture optimization:** A large number of studies have proposed to improve GNN robustness by adjusting GNN architectures. For example, a method called Gaussian-based GCN [116] introduces Gaussian distributions into graph embedding rather than using them directly. Moreover, defenders can use a graph encoder to learn the information, followed by a GAN model to conduct contrastive learning [117]. Alternatively, they can introduce meta-optimization and adapt the aggregation process or completely redesign the GNN architecture [118].

   b) **Attention mechanism:** Another effective approach involves introducing attention mechanisms into GNNs [119] to reduce the impacts of adversarial edges or nodes while maintaining the performance on the clean graph. Specifically, these methods utilize the attention scores to identify adversarial edges and nodes and then decrease their weights during the aggregation process of the GNN training. For example, a defender could assign higher scores to edges connecting more similar nodes [119].

\[
\begin{align*}
\frac{1}{m} \sum_{k=1}^{m} \frac{\partial f_\theta \left( \frac{k}{m} \times (A_{ij} - 0) \right)}{\partial A_{ij}}, & \quad \text{removing edges} \\
\frac{1}{m} \sum_{k=1}^{m} \frac{\partial f_\theta \left( A_{ij} + \frac{k}{m} \times (1 - A_{ij}) \right)}{\partial A_{ij}}, & \quad \text{adding edges}
\end{align*}
\]
c) Adversarial training: Adversarial training enhances the robustness of a model by introducing noise during GNN training [120], [121]. Specifically, defenders first use existing attack methods to construct perturbed graphs (noise) and then train GNNs on these perturbed graphs to counteract the negative impacts of potential adversarial attacks [106]. Consequently, this method is more effective in dealing with specific attacks, whose adversarial samples can be used in training, compared to other defensive methods. Furthermore, since only training graphs are modified under this approach, GNN developers do not have to modify the GNN architecture, meaning that the implementation effort required is relatively small. Nevertheless, it may be less generalizable to other types of attacks, especially those that are unknown to the developer, since adversarial samples from these attacks have not been used during GNN training.

d) Robustness certification: Robustness certification can provide certification of a GNN’s robustness under certain perturbations [122]. Specifically, it is used to quantify how node predictions from GNNs can be robust to arbitrary forms of perturbations that are bounded in a considered space. Once a node is certified, robustness certification guarantees that perturbations within the bounded space cannot change the GNN’s prediction. Recently, certification methods have been used to perform robust training of GNNs [122], [123], [124]. Defenders add an extra term to the objective function of the target GNN model, which aims at driving every node to be certified. As a result, the trained GNNs can be more robust to perturbations.

3) Defense After GNN Training: Defense mechanisms can also be implemented after the GNN training. Consequently, such defenses can be used to defend against evasion attacks that occur during the GNN inference phase.

a) Detection of malicious perturbations during inference: Another common defense approach is to detect malicious perturbations. Approaches of this kind explore the difference in inference performance between the attacked GNNs and the normal GNNs. Specifically, a recent study [125] has shown that the output scores of the malicious node predictions are significantly lower than those of clean nodes. Therefore, defenders can utilize this evidence to identify adversarial attacks. In addition, robustness certification can also be used to identify the potential target nodes by analyzing their robustness levels [122], which can help defenders detect perturbations.

4) Summary: While a growing number of methods have been designed to enhance the robustness of GNNs, they are applied in different scenarios. In the following, we compare the methods in terms of several aspects and discuss their application scenarios.

a) Phase of implementation: These defense methods are designed to counter attacks that target different stages of GNNs. As an example, pretraining defense methods should be implemented prior to training. As a result, they will be better able to defend against poisoning attacks that perturb the training graph. However, if the perturbation occurs during the inference period (i.e., evasion attacks), these approaches cannot counteract it. By contrast, posttraining defenses can detect perturbations during the inference process; however, they are unable to deal with poisoning attacks in which an inaccurate GNN model has been built and deployed for inference. Meanwhile, the during-training defense strategies [116], [119] may be able to counter both types of attacks (poisoning and evasion).

b) Modularity: Each of these defense methods has a distinct modularity. Consider an existing GNN development pipeline that needs to be updated to a robust one. When using a pretraining or posttraining defense, there is no need to alter any existing process; these methods can be implemented in a plug-and-play fashion, which can easily be incorporated into the current workflow. For during-training defenses, however, it may be necessary to make changes to the training programs.

c) Deployment compatibility: Defense methods also suffer due to varying levels of compatibility during deployment. Specifically, pretraining defenses can generate robust GNNs during their development; thus, the deployment of these GNNs does not need to be modified in comparison to the normal GNN deployment. In the meantime, during-training defense methods may require specific GNN architectures. For instance, the RGCN [116] should model the hidden layer representation of nodes as Gaussian distributions and exploit them during information propagation to reduce the impact of adversarial perturbation. Furthermore, a posttraining defense method may require the GNN systems to provide the function block for detection (e.g., comparing the output scores to a threshold [125]) in GNN systems. Because these methods may require specialized GNN architecture or additional function blocks that may not be supported by general frameworks, they are less generalizable.

d) Complexity: Despite compatibility during deployment, defending against adversarial attacks introduces varied levels of deployment complexity. This complexity is primarily driven by the computational demands inherent in either training or inference stages. Specifically, the deployment complexity might be associated with the statistics of the graph. For instance, defenses were implemented prior to GNN training, such as a training graph preprocessing method that computes the low-rank approximation of the adjacency matrix, whose complexity will be related to the training graph size [113]. Such methods necessitate additional processing blocks, thereby elevating the intricacy of
the defensive strategies. We have identified this complexity as a pivotal element in our trustworthy GNN factors, categorizing it under “environmental well-being,” and we will delve into its interactions in Section IX. Furthermore, this increased complexity can potentially limit the model’s scalability to more extensive graphs, which we will discuss in the “Future Direction” section.

C. Applications

When GNNs are widely employed in sensitive applications, their robustness becomes crucial. We highlight potential risks from adversarial attacks in several typical real-world applications to emphasize the importance of developing robust GNNs. Specifically, we introduce the case studies based on the perturbation operations mentioned in Section III-A1: perturbing graph structure/node attributes and injecting nodes.

1) Malware Detection: Given that network traffic can be represented as graph data (i.e., network traffic graphs), GNNs have become a popular tool for analyzing the traffic behavior and identifying malware [126], [127]. However, an attacker can easily modify these traffic graphs, raising concerns about the GNN’s robustness in malware detection [128], [129]. Specifically, attackers can execute various types of adversarial attacks targeting different phases of GNN development. For example, poisoning attackers can manipulate the exploitation code embedded in the malware sample and poison the training data for the malware detector if their malware samples are gathered and used by the detector developer [130]. Evasion attacks can also be executed by this approach, altering the associated traffic graph of the malware and evading detection [130]. Thus, examining potential attacks and devising countermeasures are both essential for establishing trustworthy GNNs in this domain.

2) Financial Analysis: Robustness is critical for trustworthy GNNs in financial analysis. Recently, GNNs have been deployed for credit estimation and fraud detection [131], [132]. However, the introduction of adversarial attacks can deceive the model to produce incorrect predictions. For instance, node injection attacks, which may involve inserting a few dubious transactions or creating fake accounts [133], [134], can construct the adversarial example graph and then evade the GNN model’s detection. This compromises the trustworthiness of using GNNs in the financial domain.

D. Future Directions of Robust GNNs

1) Robustness Evaluations: Despite the increasing number of studies that have proposed to develop robust GNNs, mathematically demonstrating the robustness of GNNs is a challenging proposition. In general, most robustness evaluations of GNNs are based on their defensive ability, as GNN algorithms are difficult to interpret. For example, current robust GNN techniques assess their methods by using attack deterioration, which is the decreased accuracy after an attack compared to the accuracy without attack [113], [114], [120], and defense rate, which compares the attack success rate with or without defense strategies [135]. However, the robustness conclusions obtained by these metrics are based on the specific perturbations (e.g., perturbations generated by specific adversarial attacks), meaning that general robustness against other attack algorithms cannot be guaranteed. A robustness certification attempts to offer a formal certification of robustness, but it is affected by the architecture of GNNs and the type of graph data [122], [123]. This introduces barriers when evaluating newly developed robust GNN techniques and comparing them to existing studies. It is, therefore, essential to develop a quantitative metric for evaluating the robustness of GNNs (e.g., one that identifies the upper bound of the model’s prediction divergence in a given domain [136], [137]), which can be applied uniformly across various GNN architectures, graph data [138], and different attack and defense methods.

2) Defense Scalability: Although several techniques for building robust GNNs have been developed, most of them have only been evaluated on medium-sized graphs and have not been applied to large-scale graphs in practice. This highlights the necessity of studying GNN robustness with high scalability. Researchers have recently demonstrated that GNNs in large-scale graph data can also be vulnerable to adversarial attacks [139]; furthermore, they have also demonstrated that previous attacks and robust GNN schemes are typically not scalable due to the large overhead incurred during training and inference [140]. One example concerns a revised aggregation technique [141] for robust GNN training, which involves summing up the distance matrix of neighbor node embeddings. Such an approach would increase the training effort required for large graph training [139]. In view of the wide variety of attack setups and GNN applications in existence, it is clear that the robustness analysis of GNNs applied to large-scale graphs is yet to be fully explored. It is accordingly crucial to develop more efficient and scalable defense techniques to support the building of robust GNN systems for real-world applications.

IV. EXPLAINABILITY OF GNNs

The explainability of GNNs refers to the ability to make the predictions of GNNs transparent and understandable. People will not fully trust GNNs if the predictions they make cannot be explained; this lack of trust will, in turn, limit their usage in crucial applications associated with fairness (e.g., credit risk prediction [67]), information security (e.g., chip design [142]), and life security (e.g., autonomous vehicles [75] and protein structure prediction [143]). Consequently, building trustworthy GNNs requires insights into why GNNs make particular
predictions [144], [145], [146], [147], which has driven an increase in research into the interpretability and explainability of GNNs. These abilities enable researchers to capture causality in GNNs [78] or insights for further investigation in applications [148], foster the implementation of robust GNN systems by developers [149], and guide regulators to ensure the fairness of GNNs [150].

In this section, we summarize current advancements in providing explanations for GNN predictions. We first introduce the basic concepts and categories with respect to the interpretability and explainability of GNNs. We then present some typical methodologies and conduct comparisons between them. Finally, we highlight some potential directions for future research.

A. Concepts and Categories

1) Interpretability and Explainability: Methods designed to promote machine learning interpretability can be categorized into intrinsically interpretable models and posthoc explanation methods [151]. In the context of GNNs, the interpretability of GNNs utilizes intrinsically interpretable designs [152] in GNN architectures to explain predictions of GNNs. In contrast, the explainability of GNNs aims to provide posthoc explanations after the training of GNNs. In this survey, the former kind of GNNs is called interpretable GNNs, and the latter kind of method for explainability of GNNs is called explainers for GNNs.

2) Forms of Explanations: The types of explanation results for general machine learning models include feature summary statistics, feature summary visualization, internal parameters, data points (e.g., the identification of prototypes of predicted classes), and intrinsically interpretable (surrogate) models [151]. Although there are various differences between current methods designed for GNNs, the shared concern is which edges, nodes, or features are more important to the final outputs. Thus, the explanation results of GNNs are generally expressed as graphs with certain components (i.e., edges, nodes, or features) highlighted as explanations for the current sample. Sometimes, explanations of GNNs are presented in the form of subgraphs since subgraphs with specified patterns (i.e., motifs) are the building blocks of some more complex networks [153].

3) Instance-Level Explanations, Group-Level Explanations, and Class-Level Explanations: Existing methods are categorized based on whether they provide instance-, group-, or class-level explanations for GNNs. The instance-level methods (e.g., PGExplainer [59]) provide a sample-dependant explanation for each graph sample. The group-level methods (e.g., GNNExplainer [24]) generate or choose a prototype sample as the explanation for a group of graph samples. The class-level methods use graph patterns (e.g., XGNN [154]) or statistics (e.g., OFS+OBS [148]) to explain the outputs of GNNs for specified classes. Compared with group-level and class-level methods, the explanations provided by instance-level methods are both more fine-grained and more precise since explanations for a single sample are tailored to explain the sample in question. In contrast, the class-level explanations provide a high-level insight into the behaviors of GNNs.

4) Model-Specific Versus Model-Agnostic Methods: Explanation methods for GNNs can be further categorized as either model-specific or model-agnostic based on their design. Generally, interpretable GNNs (e.g., SAN [155]) tend to be model-specific; this is because explanations are derived from specifically designed operations in GNNs, which are used to reflect the importance of certain graph components (e.g., edges, nodes, or features). Accordingly, these interpretable GNNs can usually only provide explanations for the proposed graph neural architectures. For their part, model-agnostic methods (e.g., XGNN [154]) can provide posthoc explanations for any GNN. However, not all posthoc explainers for GNNs are model-agnostic. The reason is that, like interpretations [156] for piecewise linear neural networks, some explainers (e.g., CAM [61]) are not rigorously model-agnostic since they require GNNs to employ some specified operations (e.g., global average pooling [61]).

5) Others: Methods for explaining GNNs also differ depending on the target task of the GNNs in question. For interpretable GNNs (e.g., SAN [155]), they can only provide explanations for current architectures designed for specified tasks. For posthoc explainers of GNNs, the difference on target tasks means that some explainers (e.g., GraphLime [157]) can only provide explanations for GNNs in one specified task (e.g., node classification), while others (e.g., RCEExplainer [158]) are available for multiple GNN tasks. Moreover, existing methods have different needs in terms of the prerequisite knowledge of GNNs. In this survey, the terms white-box, gray-box, and black-box are used to represent the degree of knowledge required by different methods about the target GNNs when providing explanations for them. Some explainers [154] (i.e., black-box methods) treat the GNNs as black boxes, while others (e.g., white/gray-box methods) may need to access some internal information (e.g., the backward gradients [24] and weight parameters of GNNs [61]) from GNNs to train explainers.

B. Methods

When exploring methods for explaining GNNs, two essential steps are designing the method for providing explanations and evaluating explanations. Both of these steps can be challenging for researchers. For example, due to the characteristics of graph data, traditional explanation methods for deep learning (e.g., input optimization methods [159] and soft mask learning methods [160]) cannot be directly applied to GNNs because of the irregularity and discrete topology of graphs. When conducting evaluations, domain knowledge (e.g., brain connectomics [148]) is sometimes necessary to validate GNN explanations.
To this end, various intrinsically interpretable GNNs and posthoc explainers for GNNs have been proposed. Here, we briefly introduce a typical posthoc explainer for GNNs, i.e., GNNExplainer [24]. Taking the graph classification task as an example, we here assume that a well-trained GNN model $f_\theta$ is employed as a label function to produce the predicted label $\hat{y} = f_\theta(G)$ of the input graph $G = \{A, X\}$, whose ground-truth label is $y$. GNNExplainer aims to find a subgraph $G_s = \{A_s, X_s\}$ of $G$ that can act as the explanation of prediction $\hat{y} = f_\theta(G)$ by maximizing the mutual information (MI) between $G_s$ and $y$, i.e.,

$$\max_{G_s} \text{MI}(y, G_s).$$

To solve the optimization problem on the discrete graph structure, GNNExplainer proposes to learn an edge mask $M \in \mathbb{R}^{|V| \times |V|}$ ($|V|$ is the number of nodes in $G$), and the explanation $G_s$ is calculated as follows:

$$G_s = \{A_s, X_s\} = \{A \odot \sigma(M), X\}$$

where $\sigma$ denotes the sigmoid function and $\odot$ represents elementwise multiplication. After the training, GNNExplainer can generate an explanation for the prediction of GNN model $f_\theta$.

In this section, we present common ideas behind different methods for GNN explanations. Fig. 3 illustrates an overview of the current methodologies. The bottom (i.e., green zone) of Fig. 3 shows four types of intuitions used in constructing self-interpretable GNNs. The other part (i.e., gray zone) of Fig. 3 illustrates the core modules used in five different kinds of posthoc explainers and how they interact with target GNNs (green dashed-line box) to generate explanations. Moreover, it also illustrates other typical method instances that can provide posthoc explanations for GNNs. In the following, we will introduce these methodologies and some typical instances of them.

1) Intrinsically Interpretable GNNs: The main types of self-interpretable GNNs mainly include contribution estimation, introduction of interpretable modules, embedding prototype learning, and rationale generation.

   a) Contribution estimation: Contribution estimation methods employ values from modules or operations in GNN architectures to evaluate the contribution made by particular components in input samples. These methods then utilize the estimated contributions to obtain explanations for predictions made by GNNs. In contribution-based methods, the values of contribution estimations can be derived from different modules or operations. For example, these contribution values can be provided by activation...
values (e.g., NGF [161]), attention weights (e.g., STANE [162]), or specially designed modules (e.g., SAN [155]).

b) Introduction of interpretable module: Interpretable models such as K-nearest neighbors (KNNs) can be introduced as a module in GNNs. The subsequent improvement to GNN interpretability mainly comes from changing the classification module [e.g., multi layer perceptron (MLP)] to an interpretable module, such as KNN, given that GNNs serve as encoders of graph data. This kind of architecture combines the powerful expressive ability of GNNs and the interpretability of existing interpretable methods. For example, a GNN architecture for node classification called SE-GNN [163] proposes identifying the interpretable k-nearest labeled nodes for each node to simultaneously make a label prediction and explain why this prediction has been made.

c) Embedding prototype learning: Rather than employing interpretable entities (e.g., labeled nodes in SE-GNN [163]) as in the introducing interpretable module methods, the interpretability of prototype learning methods comes from the learned embedding prototypes or embedding motifs in GNNs. For example, Zhang et al. [164] propose a framework called PortGNN that incorporates prototype projection and similarity calculation operations. The interpretability of PortGNN on a specific input graph comes from displaying the learned prototypes and similar subgraphs in the input graph. Similarly, the interpretability of another framework called MICRO-Graph [165] is supported by the learned motif embedding and node-to-motif assignment.

d) Rationale generation: In this type of method, the self-interpretable GNNs incorporate a module that can produce a subgraph rationale for the input graph data. For example, Yu et al. [166] propose a method called GIB that is based on the information bottleneck theory. It can generate subgraphs that share the most similar property (i.e., label) to the input graphs. Adopting a causal view of graph generation, Wu et al. [78] propose a framework called DIR that can produce rational subgraphs that are invariant to the changed data distribution.

2) Posthoc Explainers for GNNs: Explainers for GNNs include gradient-/feature-based methods, perturbation-based methods, surrogate methods, decomposition methods, and generation methods.

a) Gradient/feature-based methods: Gradient-based methods employ different types of gradient processing to obtain an estimation of the contribution made by components in the input samples. These methods produce local explanations for current samples by calculating the gradients of GNNs on input space. Gradient-based methods treat input samples, rather than model parameters, as variables of GNNs, and utilize the backpropagation mechanism to obtain the gradients of GNNs on the input space. For example, a method called SA [167] uses the square value of gradients to define the contribution of components (e.g., nodes, edges, or node features) in the current graph. Feature-based methods use mappings from embedding space to input space to evaluate the contribution made by components in the input samples. For example, Pope et al. [61] propose an algorithm called CAM, which maps the embedding in the last layer to the input space by means of weighted summation. The mapping weights come from the last fully connected layer used for the final prediction. The main differences between feature-based methods are the mapping construction methods employed. Moreover, according to a recent study [168], introducing regularization terms (i.e., batch representation orthonormalization and Gini regularization [168]) can improve feature-based methods, such as CAM [61] and Grad-CAM [61].

b) Perturbation-based methods: The counterfactual explanation describes causality as “if X had not occurred, Y would not have occurred” [151], [158]. In the context of explainers for GNNs, when removing one edge and node or changing one node feature from the explanation for the current output, GNNs will generate a different output if the removed or changed component is a counterfactual explanation for the current output. Accordingly, different from gradient-/feature-based methods, perturbation-based methods attempt to find the counterfactual explanation [24] or underlying cause [169], [170], [171] for GNNs by perturbing components of the graphs. Specifically, perturbation-based methods apply masks to edges, nodes, or node features to filter out the most essential components, which serve as explanations for GNNs. It is expected that the predictions of GNNs on explanations will be similar to those on input samples. For example, GNNExplainer [24] utilizes mutual information to train mask generators, which can provide explanations for both node classification and graph classification. Differences between perturbation-based methods stem primarily from their mask types, metrics for component importance, mask generation algorithms, and the forms taken by their explanations (e.g., edges, features, or subgraphs).

c) Surrogate methods: It is sometimes difficult to employ gradient/feature-based methods and permutation-based methods that require internal knowledge about GNNs (e.g., node embedding in PGExplainer [59]) because access to the GNN system may be limited (e.g., only queries are allowed). A practical solution is to employ interpretable and straightforward models (i.e., surrogates) to simulate the input–output mapping in target GNNs. By considering the similarity between the input samples and their neighbor samples, the functionally similar surrogate model can interpret the local behaviors of target GNNs to some extent. For example, a method called PGM-Explainer [172] perturbs node features to obtain similar samples and then employs an interpretable Bayesian network to provide explanations. Surrogate methods are generally model-agnostic; their differences mainly lie in the neighbor sampling strategies and interpretable surrogate models employed.

d) Decomposition methods: Unlike the above methods, decomposition methods distribute the prediction score of
GNNs on samples to the input space according to certain decomposition rules that help identify which components of the input space provide the greatest contribution to the prediction of GNNs. For example, the GNN-LRP [173] algorithm employs Taylor decomposition as rules to model interactions between GNN layers and then identifies a collection of walks to explain the output of GNNs. For decomposition methods, the decomposition rules are the main differences between them.

e) Generation methods: In generation methods, explanations for GNNs are produced directly from the explanation generator module by employing graph generation methods based on RL or other frameworks. For example, a method called XGNN [154] aims to obtain a graph pattern that maximizes the prediction of GNNs on the target class, after which this graph pattern is used as an explanation for GNNs. This approach formulates graph generation as an RL task. In each step of graph generation, XGNN makes an edge-adding decision based on the existing graph pattern. It uses policy gradients and feedback from GNNs to train the graph pattern generator. In comparison to other methods, the class-level explanation obtained from XGNN provides a high-level understanding of GNNs and a human-intelligible subgraph explanation of GNNs for graph classification. In addition to using RL frameworks (e.g., XGNN [154] and RG-Explainer [174]), other graph generation frameworks can also be employed to provide GNN explanations. For example, adopting a causal theory perspective, Lin et al. [175] propose a method called OrphicX based on the variational graph autoencoder (VGAE).

f) Others: This category contains methods that cannot be categorized into the above posthoc explanation methods. For example, Feng et al. [176] propose a method called DEGREE to estimate the contribution score of nodes and employ an agglomeration algorithm to complete the explanation subgraph construction. In DEGREE, based on the forward operation decomposition on GNN layers, the node contribution is calculated as the relative contribution (like the Shapely value [151]) when adding the target node to its contexts. Abrate and Bonchi [148] propose a heuristic-search method called OFS+OBS that finds counterfactual graphs to explain graph classification results on brain networks. Lin et al. [177] propose a framework called Gem to train an explanation generator; under this approach, the ground-truth explanations in training data are distilled through combining predefined rules and queries on target GNNs.

In addition to the above methods, some model-agnostic methods for general AI can also be extended to explain GNNs. For example, IG [178], DeepLIFT [179], and CXPlain [180] can be transferred to the graph domain to provide explanations for graph classification [171].

Remarks: After obtaining explanations of GNNs via the above methods, it is possible to evaluate the quality of these explanations through visualization and accuracy-related metrics (see Section I-C). In the Appendix, we present related resources for studying the explainability of GNNs.

C. Summary

Table 4 presents a comprehensive comparison of representative methods that provide explanations for GNNs. In the following, we also briefly compare them from several high-level perspectives.

1) Interpretability and Explainability: Interpretability and explainability refer to self-interpretable GNNs and posthoc explainers for GNNs, respectively. Generally speaking, self-interpretable GNNs can only provide explanations for themselves since the explanations are dependent on their specific architectures. Compared with most self-interpretable GNNs, posthoc explainers have a broader range of applications because they can usually provide explanations for GNNs with any architecture. However, the above difference does not mean that the functionality of self-interpretable GNNs is limited. For example, one recent GIB framework [166] simultaneously demonstrates the capability to discover the most informational subgraphs and flexibility (i.e., plug-and-play) in cooperating with other GNN backbones.

2) White/Gray/Black-Box Knowledge: When obtaining explanations from self-interpretable GNNs, users typically require to have white-box knowledge of target GNNs. Here, we compare these existing posthoc explainers based on the levels of background knowledge required.

Gradient/feature-based methods explicate GNN predictions in a straightforward manner. However, these methods usually require access to specific knowledge (e.g., inner gradients or features) about target GNNs (i.e., they treat the target GNNs as white/gray boxes). Perturbation-based methods generally adopt a causal theory-based approach and, accordingly, need to learn different kinds of masks to generate explanations. Most of them also require access to specific knowledge (e.g., backward gradients) about target GNNs to learn how to generate these masks. Compared with other methodologies, surrogate methods are more practical since they only need to query target GNNs (i.e., they treat GNNs as black boxes). In decomposition methods, explanations are derived by distributing final predictions into input space. Thus, these methods also need access to knowledge about GNNs (i.e., white-box setting) to implement the above distribution. For their part, generation methods can directly produce human-intelligible subgraph explanations for specific input graph data or target classes. Most of these approaches treat target GNNs as black boxes when training the explanation generator.

3) Reasoning Rationale: From the causal perspective, most of today’s explanation methods are based on factual reasoning (e.g., GNNExplainer [24], PGExplainer [59], XGNN [154], RG-Explainer [174], and OrphicX [175]) or counterfactual reasoning (e.g., CF-GNNExplainer [184])
and Gem [177]). One recent study [186] shows that considering only factual reasoning will result in extra information being included in explanations (i.e., sufficient but not necessary explanations), while only considering counterfactual reasoning will break the complement graph of explanations (i.e., necessary but not sufficient explanations). Existing approaches that consider both forms of reasoning (e.g., RCExplainer [158] and CF$^2$ [186]) show superiority in terms of explanation robustness [158] and quality (e.g., accuracy and precision) [186].

4) Other Limitations: In these gradient-based methods, contribution values based on gradients may only reflect the local sensitivity of GNNs; thus these methods may suffer from saturation problems [179] (i.e., the gradient of model output with respect to the inputs is zero) and explanation misleading [158], [188]. In perturbation-based methods, the continuous values in soft mask learning methods (e.g., GNNExplainer [24]) may suffer from the “introduced evidence” problem [189] (i.e., potential side effects utilized by models when introducing mask operations) [31]. Although discrete masks in some methods (e.g., ZORRO [181] Causal Screening [171]) can alleviate this issue, the explanation generation mechanism (e.g., greedy algorithms) employed therein may result in locally optimal explanations. For surrogate methods, there is no consensus on how to define neighbors of the input graph data and how to choose interpretable surrogate models [31]. Moreover, the white-box knowledge setting of decomposition methods may make them impractical for users who can only query GNNs (e.g., using cloud-based GNN services [190]). Finally, a potential weakness of existing generation methods is that they ignore counterfactual explanations (i.e., they only consider the relationship between explanations and the labels of input graph data in their reward or loss functions).

5) Complexity: Humans can understand the predictions of GNNs by deploying explanation methods. However, these deployment behaviors generally involve additional

| Methods       | Category | Methodology          | Knowledge* | Level | Task* | Form      |
|---------------|----------|----------------------|------------|-------|-------|-----------|
| NOFL [161]    | Interpretability | Contribution          | White-box | Instance | GC | Node |
| SAN [155]     | Interpretability | Contribution          | White-box | Instance | GC | Edge |
| STANE [162]   | Interpretability | Contribution          | White-box | Instance | LP/NC | — |
| SE-GNN [163]  | Interpretability | Interpretable Module  | White-box | Instance | NC | Subgraph |
| PortGNN [164] | Interpretability | Prototype             | White-box | Instance | NC/GC | Subgraph |
| MICRO-Graph [165] | Interpretability | Prototype             | White-box | Instance | GC | Subgraph |
| DIR [78]      | Interpretability | Rational              | White-box | Instance | GC | Node/Edge |
| GIB [166]     | Interpretability | Rational              | White-box | Instance | GC | Subgraph |
| SA [167]      | Explainability | Gradient/Feature-based | Grey-box   | Instance | NC | Node/Edge |
| Guided BP [167] | Explainability | Gradient/Feature-based | Grey-box   | Instance | NC | Node/Edge |
| CAM [61]      | Explainability | Gradient/Feature-based | Grey-box   | Instance | NC | Node |
| Grad-CAM [61] | Explainability | Gradient/Feature-based | Grey-box   | Instance | NC | Node |
| GNNExplainer [24] | Explainability | Perfurbation-based     | Grey-box   | Instance/Group | NC/GC | Edge/Feature |
| PGExplainer [59] | Explainability | Perfuration-based     | Grey-box   | Instance | NC/GC | Edge |
| Causal Screening [171] | Explainability | Perfusion-based     | Grey-box   | Instance | NC | Node/Feature |
| GraphMask [182] | Explainability | Perfuration-based     | White-box | Instance | SRL/MQA | Edge |
| SubgraphX [183] | Explainability | Perfuration-based     | Black-box | Instance | NC/GC | Subgraph |
| CF-GNNExplainer [184] | Explainability | Perfuration-based     | Grey-box | Instance | NC | Edge |
| RCEExplainer [158] | Explainability | Perfuration-based     | Grey-box | Instance | NC/GC | Edge |
| ReFine [185]  | Explainability | Perfuration-based     | Grey-box | Instance | GC | Edge |
| CF$^2$ [186]  | Explainability | Perfuration-based     | Grey-box | Instance | NC/GC | Edge/Feature |
| GraphLine [157] | Explainability | Surrogate             | Black-box | Instance | NC | Feature |
| Relief [187]  | Explainability | Surrogate             | Black-box | Instance | NC | Edge |
| PGM-Explainer [172] | Explainability | Surrogate             | Black-box | Instance | NC/GC | Node |
| LRP [167]     | Explainability | Decomposition         | White-box | Instance | NC/GC | Node/Edge |
| Excitation EB [61] | Explainability | Decomposition         | White-box | Instance | GC | Node |
| GNN-LRP [173] | Explainability | Decomposition         | White-box | Instance | GC | Walk |
| XGNN [154]    | Explainability | Generation            | Black-box | Instance | GC | Subgraph |
| RG-Explainer [174] | Explainability | Generation            | Black-box | Instance | NC/GC | Subgraph |
| OrphicX [175] | Explainability | Generation            | Grey-box | Instance | NC/GC | Subgraph |
| DEGREE [176]  | Explainability | Others                | Black-box | Instance | NC/GC | Node/Edge |
| OFS-OBS [148] | Explainability | Others                | Black-box | Instance/Class | GC | Edge/Subgraph |
| Gem [177]     | Explainability | Others                | Black-box | Instance | NC/GC | Edge/Subgraph |

* In the Knowledge column, “White/Grey/Black-box” indicate that the target GNN is treated as a white/grey/black box when obtaining explanations on it. The “GC”, “LP”, “NC”, “SRL” and “MQA” in the Task column represent graph classification, link prediction, node classification, semantic role labeling, and multi-hop question answering, respectively.
effort (e.g., the training of posthoc explainers), which makes it necessary to consider the complexity of explanation methods when implementing them. First, the space complexity of an explanation method can be potentially affected by the graph whose prediction needs to be explained. For example, the parameter size of GNNExplainer is independent of the graph size [59]. Second, the time complexity differs among different explanation methods. For instance, given a graph with $E$ edges, the GNNExplainer [24]) owns $O(T|E|)$ time complexity, where $T$ indicates the epoch number when training the GNNExplainer [59]. However, that of PGExplainer is $O(|E|)$ once it is well trained. The time complexity difference makes the PGExplainer more efficient in generating explanations for new prediction instances [191]. Note that the complexity of deploying GNN explanation methods also includes its influences on other aspects (e.g., robustness and environmental well-being) of trustworthy GNNs. Refer to Section IX for more details.

D. Applications

By providing insights into why GNNs make particular predictions, explainability facilitates the verification of prediction to protect users from potential adverse effects when accepting predictions without scrutiny, which fundamentally improves the trustworthiness of GNN systems. Next, we present representative applications where the trustworthiness of GNN systems can be greatly improved by explainability.

1) Medical Diagnosis: When GNNs are deployed in medical systems (e.g., breast cancer subtype classification [192] or histopathology analysis [193], [194]), prediction trustworthiness can be improved by involving GNN explanation methods. In clinical diagnosis, representing each patient as a protein–protein interaction network, a variant of the GNNExplainer provides explanations to further verify the disease (e.g., cancer) detection results [195]. By establishing causal relationships between multigranularity features and diagnosis results, a method called CMGE extracts the entities most relevant for diagnosis from electronic medical records [196]. In computational pathology [197], posthoc GNN explainers provide intuitive pathological entity graphs to explain Cell-Graph representations for breast cancer subtyping [62], enhancing transparency in clinical decisions [198].

2) Guilty Verdict: As social networks [199], [200], [201] are emerging in the current society, the explainability of GNN can provide evidence for the prediction of illegalities of users on social networks. For example, in the detection of fake news, a method called GCAN uses the distribution of attention weights to reveal what words and which users are vital for the detection results [200]. When GNNs are employed in police systems, the GNN explainability benefits antidrug police by providing evidence for the prediction of drug abusers on social networks [199].

Remarks Note that explainability also helps to verify the prediction in other GNN applications, such as cyber security [202], [203], [204] and natural language processing [182]. Furthermore, GNN explainability contributes to the improvement of GNN performance [205], [206] and the discovery of new knowledge [148], [168], [207], [208], [209], [210].

E. Future Directions of Explainable GNNs

1) Strictly Model-Agnostic Methods: Compared with self-interpretable GNNs, one advantage of posthoc explainers for GNNs is their ability to explain GNNs with various architectures. However, only a few existing explainers are strictly model-agnostic (i.e., black-box in Table 4) methods (see Section IV-C). Of the existing explainers, the methods that require black-box knowledge of GNNs (see Section IV-C) present stronger practicality, as users sometimes cannot obtain enough knowledge about GNN systems to support a white-box approach. For example, when using cloud-based GNN services [190], users can only query GNN services to obtain predictions on the input graph data and cannot access the inner backward gradients of GNNs, which are indispensable for the training of some explainers (e.g., GNNExplainer [24] and PGExplainer [59]). Due to the diverse application scenarios of posthoc explainers, designing compelling and low-demand explainers represents a promising avenue for facilitating the explainability of practical GNN systems.

2) Evaluation Benchmark for Real Applications: The absence of real-world ground-truth datasets hampers the exploration and identification of practical GNN explanation methods. Existing explanation methods are generally evaluated on synthetic datasets with visualization and accuracy-related metrics [31]. Although some methods, such as GNNExplainer, obtain competent performance on these datasets [24], they have limited explanation accuracy for GNNs on real-world datasets such as scene graphs [171]. Therefore, designing evaluation datasets and metrics for different applications is crucial for boosting GNN explanation methods and comprehensively evaluating their performance. Another possible solution is to explore the integration of visual analytics and interpretable GNNs, which can facilitate explanation evaluation in real-world applications.
In this section, we present recent studies that have examined privacy issues related to GNNs. First, we introduce some privacy-related attacks that reveal the privacy risks in GNN systems. We next discuss several privacy-preserving techniques used in GNN systems. Finally, we provide an outlook on the future directions in GNN privacy.

A. Privacy Attacks

GNN systems face the threat of data privacy breaches. Through the exploitation of certain vulnerabilities, several privacy attacks have been proposed with the goal of inferring sensitive information from GNN systems. Specifically, there are two main factors to consider when analyzing these attacks: attack targets and attack knowledge.

1) Attack Targets: Attackers attempt to steal the private information contained in GNNs, including models and graph data.

   a) Models: GNN models, including both model architectures and parameters, often represent the intellectual properties of model owners. For example, an e-commerce company may employ a confidential GNN for recommendation [212], [213], making it an important commercial property and revealing model rules will have grave privacy implications.

   b) Graph data: Graph data, which also contain a high degree of sensitive information, should be kept private. For example, patients expect to use GNNs for medical diagnosis without their personal medical profiles being leaked [214], [215]. Note that, in the graph data context, “privacy” refers to not only the privacy of the graph samples but also the graph statistical properties and training membership (i.e., whether or not a node/edge/graph sample is engaged in GNN training).

2) Attackers’ Background Knowledge: Attackers are assumed to have access to different knowledge about target GNNs.

   a) Black-box knowledge: In black-box settings, attackers can only send queries to GNN models but cannot access the training graphs and inner knowledge of GNN models [47], [216]. For example, in the context of machine learning as a service, attackers have a similar status to end-users in that they only have access to the public APIs of the models.

   b) White-box knowledge: Attackers can also obtain white-box access to the target GNN models in certain scenarios [216], [217]. Note that, unlike the setting of adversarial attacks in Section III-A1 (where the entire training process is known), white-box attackers in this context can only access GNN models, while the training graphs and labels remain unknown. This scenario often arises when model owners publish their GNNs while still wishing to keep their training data and training strategies secret.

In this section, we summarize existing privacy attacks and introduce them based on the abovementioned aspects.

3) Model Extraction Attacks: Model extraction attacks are designed to steal information about the architecture and parameters of a GNN model. Specifically, they can reconstruct the original model or construct a substitute model that performs similar to the original. Current model extraction attacks on GNNs focus only on node classification tasks [63]; in comparison, there is little research on attacks against graph classification and link prediction. The attack methods are designed based on different background knowledge of the attackers. For example, knowing the node attributes of a set of adversarial nodes enables attackers to use discrete graph structure learning methods (e.g., LDS [218]) to construct a connected substitute graph based on these node attributes. The attackers can then generate queries from these adversarial nodes and use the responses to train a substitute GNN [63].

4) Membership Inference Attacks: Membership inference attacks aim to infer whether a specific component or sample has been included in the training dataset of a victim GNN. These attacks can be categorized depending on the GNN target task involved.

   a) Node-level attacks: In node-level classification tasks, membership inference attacks aim to determine the membership of a specific node. Specifically, attackers intend to determine whether a node has been used as a training node for GNNs [48], [216].

   b) Link-level attacks: As representations of nodes’ relationships, edges also contain private information and have, thus, become common targets of membership inference attacks. This type of attack attempts to determine whether a specific link between two nodes exists in the training graph [47]. Since GNNs essentially aggregate information about each node from its neighbors, the output posteriors of two connected nodes are likely to be closer together than those of non-connected nodes. Therefore, using only the posteriors of nodes obtained from the target model, the attackers can calculate the distance between posteriors of a node pair and select a threshold to determine the membership of the links.

   c) Graph-level attacks: Graph-level membership inference attacks are designed to identify the membership of an entire graph rather than the individual components (i.e., a single node or edge) within it. Recent work [49], [219] has demonstrated that graph-level GNNs with several popular architectures are vulnerable to membership inference attacks.

5) Model Inversion Attacks: Model inversion attacks aim to extract information about model inputs from their
corresponding outputs. Specifically, attackers might exploit a GNN’s outputs (e.g., node embedding and graph embedding) to obtain sensitive information about input graphs (e.g., node attributes or graph properties). Two popular types of inversion attacks are introduced here based on the information that they attempt to recover:

a) **Property inference:** Such attacks attempt to infer basic properties of the target graph (i.e., the number of nodes, edges, and graph density) from the graph embedding [217].

b) **Graph reconstruction:** This type of attack aims to directly reconstruct the original graph or attributes based on the output embedding. For graph structure reconstruction, given a targeted GNN and some other knowledge (such as node labels and attributes), attackers can recover a specific subgraph or the entire graph [216], [220].

6) **Other Privacy Attacks:** In addition to the above attacks, researchers also exploit privacy leakages from other attack surfaces. For example, considering GNN systems with a vertical input graph partition (e.g., node features and edges are held by different individuals), attackers who know the node features can infer the private edges held by other partitions [221]. Another type of attack involves inferring the node labels from a link prediction GNN, in which node labels are supposed to be hidden from users [64].

7) **Summary:** The attacks discussed above consider different targets and, thus, have different design rationales. Model stealing and graph property inference are primarily based on the strong relationship between the query input, the output, and the learned GNN models. The model extraction attack, for example, reconstructs a model based on the mapping between input and output. Property inference is also based on this mapping but for attribute inference. Alternatively, membership inference attacks make use of the fact that the GNN model remembers the training data, commonly identifying the difference between members and nonmembers based on the output posteriors alone.

Unlike privacy attacks on DNNs, these privacy attacks on GNNs may benefit from the unique properties of graphs and GNNs. As an example, the effectiveness of stealing links [47] (membership inference attacks targeting an edge in the training graph) exploits the similarity between the outputs of two connected nodes. Moreover, a model extraction attack [63] simulates the propagation of information in GNNs in order to construct a surrogate graph.

### B. Privacy-Preserving Techniques for GNNs

In light of the privacy threats associated with GNN systems, several privacy-enhancing techniques for these systems have been developed. In this section, we will introduce several of the most popular approaches.

#### 1) Federated Learning:

FL [222] is a popular paradigm that enables individuals (e.g., mobile devices) to train ML models collaboratively without revealing each individual’s raw data. In FL, the data belonging to these decentralized individuals can be considered private, so it is processed and trained locally without being shared with others. These decentralized individuals’ data are considered private, processed, and trained locally without sharing with others. To facilitate privacy-preserving training, a server continuously gathers and aggregates parameters (e.g., gradient/model weights) from each individual until the model performance converges. Recently, FL has also been increasingly investigated and applied in the GNN context. Table 5 summarizes several existing studies about FL in GNNs. In particular, based on how the graph is distributed to individuals, we introduce two types of FL in the GNN context:

- **Intergraph FL:** Intergraph FL [223] can be considered as a natural extension of DNN-based FL tasks (e.g., FL for image classification). For this particular type, each individual processes its own set of local graph samples, performs training collaboratively, and participates in producing a global GNN coordinated by a central server. A typical application of intergraph FL can be found in the biochemical industry, where each pharmaceutical company possesses a confidential dataset consisting of a collection of molecule graphs. These companies perform intergraph FL to train a global GNN for drug property analysis without the need to pool their confidential data into a central server [224].

- **Intragraph FL:** Intragraph FL is designed for scenarios in which multiple clients want to jointly train a GNN on a global graph, with each of them owning a local subgraph of this global graph [223], [225]. Based on the distribution characteristics of the global graph
(i.e., how the entire graph is distributed to each client in the feature and node space), there are two types of intragraph FL [226].

i) **Horizontal intragraph FL:** This type considers the case in which the local (sub)graphs belonging to each individual are horizontally partitioned. In this case, a global graph is dispersed over multiple individuals, each of whom owns partial nodes and is interested in training collaboratively without data leakages [33], [227]. For example, in social networking apps, each user has a local social network, which is a subgraph of the overall human social network [228]. Through the use of horizontal intergraph FL, a global GNN for recommendation can be generated while protecting the private graph data of each of these instances.

ii) **Vertical intragraph FL:** This type of FL considers a scenario in which the subgraphs owned by each individual are vertically partitioned. Specifically, the subgraphs of individuals are distributed in feature space; while each of them possesses a different feature and label space, they all share the same nodes and their connections. The FL method is commonly used to train global GNNs for multiple cooperative organizations. For instance, to develop a graph-based financial fraud detection system [229], banks and regulators with common customers (e.g., Webank [230]) and the National VAT Invoice Verification Platform [231] opted to train a GNN collaboratively; these companies share the same nodes (common customers), but each has its own distinct feature space (loan records in different institutions) [232].

Similar to FL in DNNs, FL in GNNs protects the private data of each individual by transmitting model parameters rather than raw data between the individuals and the central server during training. In this way, private data can be kept local and cannot be accessed by others. Specifically, each individual can train a GNN model with its own data. Subsequently, they can upload their local model to the server, which then performs an aggregation function (such as FedAvg [281]) to build a global GNN model. Since such aggregation uses GNN parameters inductively (i.e., the model is independent of its structure), topological information about the local graph data is not required and can accordingly be kept private.

Unlike DNN-based FL tasks, FL in GNNs often incorporates carefully designed mechanisms. Due to the characteristics of graph data, certain challenges that arise during the training or implementation of FL systems may manifest or be amplified in the GNN context. These challenges can be summarized as follows.

i) **Non-i.i.d. individual data:** Similar to ordinary DNNs, FL in GNNs also relies on stochastic gradient descent (SGD), which makes their training performance easily affected by non-i.i.d. training data [248]. In practice, due to the heterogeneity of the structures and features of the graphs owned by different individuals, non-i.i.d. individual data (i.e., diverging graph structure distributions and node feature distributions of the local graph data) are an inevitable problem for FL with graphs [235]. Thus, most FL methods designed for GNNs aim to to minimize the impact of non-i.i.d. data on the training performance. As an example, ASFGNN [228] decouples the FL training and refines the global model parameters based on the Jensen–Shannon (JS) divergence to ensure that the loss computation is not biased by the individual data. In GraphFL [227], a metalearning technique called model-agnostic metalearning is employed, which has been proven to deal efficiently with non-i.i.d. data.

ii) **Graph isolation:** In horizontal intragraph FL, local graph data can be considered as subgraphs isolated from a latent global graph. Since representation learning on graph models relies on messages passing through the connections, private information from the other subgraphs cannot be gathered, which will impact the accuracy of the GNNs. Existing FL methods designed for GNNs address this problem by transmitting the higher domain representation rather than the raw connections [245], [246] or by using a generator to generate missing neighbors across distributed subgraphs [244].

2) **Differential Privacy:** DP, as a well-known technique for privacy-preserving ML algorithms, can guarantee that an attacker will be unable to derive private information regarding a specific piece of training data from a released learning model, with high confidence [249]. Providing such a guarantee of privacy is also a crucial requirement when developing privacy-preserving GNN applications. As an example, a social smartphone application server stores information on social interactions between its users. The server may, however, also want to utilize users’ private data, such as lists of installed applications or usage logs, in order to develop better GNN models and provide better services (e.g., recommendation system) [250]. Due to privacy concerns, the server should not access users’ raw data without any data protection.

To solve this problem, the key idea behind DP is that rather than disclosing their private data, data owners are advised to add noise to their data before sending them to the server. The data are perturbed in such a way that they will appear meaningless when viewed individually but approximate the analytics result when aggregated [250]. In prior studies of applying DP to machine learning algorithms using SGD [249], [251], perturbation is added to the gradients generated during the model training. In GNNs, the perturbations can be applied to node features [250], [252]. For example, consider a case in which DP is used in a node classification training task [250]. In this task, a server aims to learn a GCN model that can
classify nodes without knowing the private node features of the users. Users can add noise to their private feature $X_{i,d} \in \mathbb{R}$, which denotes a feature value of the $d$th dimension in $X$ for node $v_i$, so that the perturbed features $X'_{i,d} \in \mathbb{R}$ satisfy the following conditions:

$$\mathbb{E}[X'_{i,d} - X_{i,d}] = 0.$$ 

When the server performs a mean aggregation (which can also be extended to other aggregation functions), for any node $v \in V$ and any feature dimension $d \in \{1, 2, \ldots, D\}$

$$\begin{align*}
M_{v,d} &= \frac{1}{\mathcal{N}(v)} \sum_{u \in \mathcal{N}(v)} X_{u,d}, \\
M'_{v,d} &= \frac{1}{\mathcal{N}(v)} \sum_{u \in \mathcal{N}(v)} X'_{u,d}.
\end{align*}$$

Thus, based on Bernstein inequalities, it satisfies the following:

$$\Pr \left( \|M'_{v,d} - M_{v,d}\| \geq \lambda \right) \leq \epsilon$$

where $\Pr$ is the probability and $\lambda$ and $\epsilon$ are hyperparameters used for adjusting the utility and privacy. As a result, the server can use these approximations of aggregation results generated from perturbed inputs and achieve privacy preservation.

3) Insusceptible Training: Some other studies [65], [253], [254] have attempted to defend against privacy attacks and reduce the leakage of sensitive information via modifying the training process of GNNs. For example, it is possible to add privacy-preserving regulation items in the loss function during GNN training or introduce privacy-preserving modules in GNN architectures to reduce privacy leakage [65]. Specifically, consider a defender aiming to defend against a private attribute inference attack $\mathcal{F}_A(v_i)$. The defender can modify the original objective functions by adding a privacy leakage loss during the target GNN training, as follows:

$$\min_{\theta} \sum_{v_i \in V} \mathcal{L}_{\mathcal{Y}}(f_{\theta}(v_i)) + \lambda \mathcal{L}_{\mathcal{A}}(\mathcal{F}_A(v_i))$$

where $\mathcal{L}_{\mathcal{Y}}(\cdot)$ and $\mathcal{L}_{\mathcal{A}}(\cdot)$ are the loss function of the target GNN utility and attribute inference attack, respectively. Other types of defenses propose to filter out the sensitive attributes by learning GNN encoders [253], [254].

4) Security Computation: Security computation has also been widely adopted to protect data privacy in general data analytics systems and services. There are three main types of techniques for security computation, including trusted executive environment (TEE) [255], [256], homomorphic encryption (HE) [257], [258], and multiparty secure computation (MPC) [259], [260], [261]. Note that since the underlying operations (e.g., matrix multiplication) in GNNs are similar to those in ordinary DNNs, most of the existing security computation methods in DNNs can be directly extended to GNNs.

5) Summary: We examine and compare the above privacy-preserving methods from three perspectives:

a) Utilization contexts: Insusceptible training is designed to protect one type of sensitive information (e.g., a specific sensitive attribute). As a consequence, it cannot protect against other privacy threats that target other types of private information [253], [254]. Meanwhile, FL seeks to protect local data in a distributed learning system [33], [233]. Local users will always hold their personal data, meaning that the privacy of all their local data (rather than certain attributes only) can be guaranteed. Moreover, FL is designed for collaborative model training, meaning that it cannot directly protect against attacks that occur during the inference process (e.g., model extraction attacks and membership inference attacks) [253]; by contrast, insusceptible training can prevent the leakage of private information during inference periods.

b) Tradeoff between privacy, accuracy, and efficiency: As previous studies have shown, there is a tradeoff between privacy, accuracy, and efficiency [249], [250]. For example, the introduction of DP into a GNN will reduce its overall accuracy. One recent study using DP examined this tradeoff between privacy and accuracy [250], [252]. On the other hand, these privacy-preserving techniques also lead to a higher time expenditure. In the context of FL, multiple rounds of communication among different individuals greatly increase the time cost of completing the model training [222], [262]. Moreover, a more critical bottleneck is the increase in latency during inference when using security computation techniques. In the case of privacy-preserving inference using HE and MPC, the inference latency will increase significantly, becoming orders of magnitude higher than the latency of computation in cleartext [257], [263].

c) Complexity: Incorporating privacy-preserving techniques into GNNs brings inevitable additional costs, manifesting as higher computational and communication complexities. Here, we discuss the complexity of three types of privacy-preserving techniques that we mentioned in Section V-B.

i) Higher computational complexity in DP: Training with DP may require more computational effort. For instance, in [250], additional hops’ aggregation (rather than neighbor aggregation) is introduced to deal with the noisy aggregation results for low-degree nodes.

ii) Higher communication complexity in FL: FL heightens communication complexity due to the necessity for additional communication rounds since
they train the model by conducting local training and subsequently aggregating to a global model in iterative rounds [222]. Moreover, since GNNs typically derive graph embeddings from messages aggregated from neighboring nodes, vertical intragraph FL might necessitate additional communication regarding neighborhood information among connected subgraph clients for achieving better model performance [245], [264].

iii) Higher computational and communication complexity in security computation: Techniques such as secure inference introduce additional complexities in both perspectives. For example, employing encryption [257], [258] increases computational demands, while MPC methods [259], [260] simultaneously call for additional communication rounds among multiple parties and more complicated computational steps. Note that while secure computation design in DNNs focuses more on the complexity posed by secure nonlinear activation functions, GNNs introduce distinct challenges. In GNNs, the complexity arising from secure neighboring aggregation cannot be overlooked [265], [266], [267]. Encrypting computation inputs, such as the adjacency matrix and node attribute matrix, can be especially costly for graphs with a vast number of nodes [265], [267].

These higher complexities invariably influence GNN efficiency, leading to a discernible tradeoff among privacy, accuracy, and efficiency mentioned above. Furthermore, similar to the complexities observed in robust GNNs, we consider the complexity of privacy-preserving GNN techniques as a factor in environmental well-being. The environmental impact of these challenges will also be discussed in the interaction section (see Section IX).

C. Applications

Privacy concerns have been heightened with the increased utilization of GNNs in domains with sensitive data. In this section, we discuss the privacy risks in several applications and emphasize the necessity of factoring in privacy when aiming to build a trustworthy GNN. Specifically, we discuss their privacy consideration based on two key objectives of the privacy focus mentioned in Section V-A: the GNN model and the graph data.

1) Medical Diagnosis: GNNs have gained traction in drug discovery as they can naturally represent protein–protein interaction networks, providing invaluable training data for GNN models [9], [199], [268]. Such training graph data often stems from costly experiments and holds intellectual property rights. Therefore, it is imperative to safeguard this training data against membership inference attacks. Furthermore, there is also a need for stringent protection of graph data used for inference. A case in point is the recent application of GNNs for inferring COVID-19 infections [269]. Such GNN applications encapsulate data about interactions between confirmed cases, encompassing contact times and individual properties—all of which are sensitive. Therefore, it is paramount to shield these data from inference attacks to maintain the trust of data contributors, especially when using GNNs for such inferences.

2) Recommender Systems: For companies that own GNN models, there is an inherent privacy risk associated with the models themselves, such as defending against model extraction attacks. Modern recommendation systems have integrated GNNs into their architecture [270], [271], [272]. These systems are trained on vast datasets and are often deemed as valuable commercial assets for businesses. Unauthorized access or theft of these models could result in substantial losses for companies. As such, there is an urgent need to comprehensively understand the privacy vulnerabilities of GNN models and implement measures against privacy breaches. Only then can we cultivate trustworthy GNNs that businesses can confidently adopt, knowing that their privacy concerns are addressed.

D. Future Directions of Private GNNs

1) Leakage From Gradient: The risks caused by leakage from gradient [273] have not been fully explored in the GNN context. Specifically, most current works focus on discussing the potential risks caused by the leakage of final prediction results (such as prediction labels [63] and confidence scores [47]). However, final results are not the only information that can give rise to privacy leakage; an attacker can also infer sensitive information from other types of knowledge. It was recently determined that leaking gradients containing training or inference information can also lead to serious privacy issues [273]. In the case of FL, for example, attackers can reconstruct the private training data of local clients through the sharing gradient, which is also known as a gradient inversion attack [274], [275]. Further privacy analysis on how leakages of gradients can harm GNN privacy should be thoroughly explored.

2) Defense Against Privacy Attacks: Despite the fact that several privacy-preserving techniques are available for GNNs, not all of them are suitable for defending against privacy attacks. Specifically, many privacy attacks operate under black-box settings in which attackers can only send black-box queries to target GNNs [47], [216]; in short, they can infer sensitive information without knowing the internal operations of a target GNN system. It should be noted that many privacy-preserving GNN techniques keep graphs or GNN models secret during the process of training or inference [228], [233]; they are not designed to prevent the information leakages revealed by the intended results (e.g., trained GNN models or prediction results). Accordingly, they do not provide formal privacy guarantees with regard to the indirect leaks of information when an attacker attempts to infer sensitive information through privacy attacks. As an example, FL on GNNs offers a method for protecting private data by storing it locally on the client side and training GNNs based only on gradients uploaded by clients [226], [244]. After GNN training is
complete, attackers can still infer membership of elements in the graph data from the final well-trained model [47], [216]. It is, therefore, necessary to further explore how to build a privacy-preserving GNN system that is able to defend against these privacy threats.

VI. FAIRNESS OF GNNs

Fair systems win people’s trust by protecting the vital interests of vulnerable groups or individuals. In the context of GNNs, fairness means that prejudice or favoritism toward an individual or a group is excluded from GNN predictions. As the generalization of deep neural networks on the graph domain, current GNNs are data-driven and designed to learn desired mappings from graph data. It should be noted here that elements of the GNN learning process, such as message-passing [see (1)], can amplify historical or potential bias in graph data, which results in discrimination or bias in predictions of GNNs with respect to sensitive attributes (e.g., skin color) [23], [276].

In this section, we summarize current efforts targeted at achieving fairness in GNNs. We first introduce definitions and the metrics commonly used to measure fairness. Next, we present the common ideas underpinning existing methods. Finally, we discuss future directions in exploring fairness enhancements for GNNs.

A. Concepts and Categories

1) Bias, Discrimination, and Fairness: Bias and discrimination are two common concepts that are related to unfairness in AI-related research; unfairness stems from both of these factors [43]. In the pipeline of AI systems, the main types of bias are data to algorithm bias, algorithm to user bias, and user to data bias [43]. Bias comes from unfair operations in data collection, sampling, and measurement, while discrimination is caused by intentional or unintentional human prejudices and stereotyping with regard to sensitive attributes (e.g., race) [43].

Fairness is an essential and indispensable factor in maintaining social order. The disciplines of philosophy and psychology have a long history of fighting against bias and discrimination, dating back to well before the birth of computer science [43]. However, owing to the diversity of cultures and application contexts involved, there is no universal definition of fairness. Fairness can be broadly defined as “the absence of any prejudice or favoritism toward an individual or a group based on their intrinsic or acquired traits in the context of decision-making” [43], [277].

2) Unfairness Cycle: Bias emerges from biases in data, algorithms, and user interactions [43]. As shown in Fig. 4, during the generation of graph data (e.g., annotation [10]), human behavioral bias or content production bias can, in turn, introduce bias into graph data. For example, in Pokec (a popular social network in Slovakia), the number of intragroup edges far exceeds the number of intergroup edges [23]. During the learning of GNNs, due to the operation bias (i.e., the aggregation that only considers one-hop neighbors) based on graph structures, the message-passing mechanism [see (1)] in GNNs can enlarge and perpetuate existing data bias by stacking graph filtering operations [23]. Moreover, some algorithmic operations (e.g., random walks) can also introduce unfair behaviors into the learning process [278]. Upon completion of deployments, these GNNs offer unfair recommendations to users. Subsequently, after receiving recommendations, users will interact with them according to their personal preferences. The behavioral biases of different users will further aggravate the bias in graph data.

3) Group Fairness, Individual Fairness, and Counterfactual Fairness: In the context of GNNs, the three types of fairness most commonly discussed are group fairness, individual fairness, and counterfactual fairness. Group fairness requires that different groups are treated equally by GNNs. A typical definition is demographic parity [279], which requires that the predictions \( \hat{Y} \) of GNNs satisfy

\[
Pr(\hat{Y} \mid S = 0) = Pr(\hat{Y} \mid S = 1)
\]

where \( Pr \) is the probability and \( S \) indicates whether the sample comes from the protected (e.g., female) group [280]. Other common definitions of group fairness include conditional statistical parity, equalized odds, equal opportunity, treatment equality, and test fairness [43].

Individual fairness requires that similar individuals obtain similar predictions from GNNs. A typical description is the \((d_1, d_2)\)-Lipschitz property [35], [281], [282]

\[
d_1(f(x), f(y)) \leq Ld_2(x, y) \quad \forall (x, y)
\]

where \( f(x) \) is the output of GNNs on sample \( x \), \( L \) is a constant scalar, and \( d_1(\cdot, \cdot) \) and \( d_2(\cdot, \cdot) \) are similarity functions for the output and input spaces of GNNs, respectively. The \((d_1, d_2)\)-Lipschitz property can also be used to measure the
stability of $f(\cdot)$ when faced with slight perturbations in input samples.

Counterfactual fairness requires GNNs to make the same predictions for “different versions” of the same candidate [283]. It can be formalized as follows:

$$\Pr(\hat{Y} \mid s = s_1, x = x) = \Pr(\hat{Y} \mid s = s_2, x = x)$$

where condition $s = s_1, x = x$ indicates version 1 of $x$ with sensitive attribute $s_1$, and condition $s = s_2, x = x$ indicates version 2 of $x$ with sensitive attribute $s_2$.

4) Processing Stages: Based on the stage in which the methods are engaged, methods for achieving fairness can be categorized as preprocessing, in-processing, and postprocessing methods [43]. Preprocessing methods (e.g., debiasing the input graph in InFoRM [35]) attempt to remove the potential bias in the training data. In-processing methods (e.g., REDRESS [284]) modify the operations or objective functions of algorithms in training to avoid unfairness. Postprocessing methods (e.g., debiasing results in InFoRM [35]) transform the outputs of models to achieve fairness in the results.

5) Explainable and Unexplainable Discrimination: Discrimination can be subdivided into explainable discrimination and unexplainable discrimination. If discrimination is explainable, this means that “differences in treatment and outcomes among different groups can be justified and explained via some attributes in some cases” [43]; thus, explainable discrimination is considered to be legal, reasonable, and acceptable. For example, although the annual average income of females in the UCI Adult dataset is less than that of males [285], this is acceptable since males work more hours per week on average than females. On the other hand, unexplainable discrimination is unacceptable since it cannot be justified with reference to acceptable reasons. The above discrimination categories commonly used in machine learning help to clarify the research goals of GNN fairness. In this survey, we focus on the latter kind of discrimination and the methods employed to alleviate it.

Unexplainable discrimination can be further subdivided into direct discrimination and indirect discrimination [43]. Direct discrimination occurs when some sensitive and protected attributes cause the undesirable results. For example, race, skin color, national origin, religion, sex, familial status, disability, marital status, recipient of public assistance, and age are all protected attributes under the Fair Housing and Equal Credit Opportunity Acts [286]. Moreover, indirect discrimination refers to instances in which individuals are treated unjustly as a result of certain seemingly neutral and nonprotected attributes in some sensitive applications. For example, the residential zip code, which may be related to race [287], can be employed as an essential factor in the loan decision-making process.

6) Productive Bias, Erroneous Bias, and Discriminatory Bias: Productive, erroneous, and discriminatory bias are concepts that have been proposed to add some nuance to the much-abused “bias” [10]. Productive bias indicates that an algorithm is biased toward certain distributions or functions [288], on which its performance is better. This conforms to the “no free lunch theory” [289]. Erroneous bias is the systematic error caused by faulty assumptions [10]. It indicates that the undesirable outputs of algorithms are caused by some biased operations (e.g., sampling bias), which cause the model to deviate from the assumptions (e.g., distribution consistency between training data and real data). Finally, discriminatory bias is used to describe an algorithm’s unfair behaviors targeting a certain group or individual [10].

GNN fairness can be further clarified with reference to the above bias categories used in AI. Productive bias is beneficial and acceptable, like explainable discrimination. Erroneous bias generally goes unnoticed by humans since the violation of assumptions is difficult to measure. In this survey, we focus on the methods for solving unfairness under the scope of discriminatory bias, which, like unexplainable discrimination, is unjustified and unacceptable.

B. Methods

Current methods that aim to achieve fairness in the GNN context can be categorized into fair representation learning methods and fair prediction enhancement methods.

Fair representation learning is a typical method for achieving fairness in GNNs. In fair representation learning, sensitive information (e.g., gender) is excluded from learned representation to ensure the fairness of GNN predictions in downstream tasks. Within GNN architectures, adversarial learning is employed to achieve fair representation, and the goal of adversary modules is to infer sensitive attributes from learned representation. Fair representation is achieved once adversary modules are unable to infer these attributes. Two typical instances of fair representation learning are compositional filters (CFs) [290] and FairGNN [23].

Fair prediction enhancement methods aim to achieve fairness in GNNs by introducing extra processes or operations into the GNN pipeline. Depending on the type of fairness involved, these methods can be categorized as group fairness, individual fairness, or counterfactual fairness enhancement methods. Current efforts utilize different strategies to eliminate bias in GNNs and, thereby, achieve fair predictions. These strategies include data augmentation methods [67], fair graph methods [34], and regularization methods [35].

Under the data augmentation strategy, the counterfactual fairness on sensitive attributes can be regarded as invariant to the sensitive attribute values. Similar to adversarial training for robustness, the data augmentation strategy obtains “adversarial samples” by perturbing these protected attributes and then adds them into the GNN
training data to achieve “robust” (i.e., fair) predictions on nodes. One representative example of data augmentation is a method called NIFTY [67]. The fairness in NIFTY is defined as counterfactual fairness, in which the encoder ENC satisfies the following:

\[ ENC(u) = ENC(\tilde{u}^s) \]

where \( \tilde{u}^s \) is obtained by modifying or flipping the sensitive feature \( s \) of node \( u \). To achieve fairness in GNNs, NIFTY utilizes the Siamese learning approach and the augmented view of attribute information.

Fair graph methods aim to implement fairness-oriented modifications on the graph structure or node features before or during the training of GNNs. From the perspective adopted by these methods, biased predictions made by GNNs stem from unfair connections in the graph structure or biased node features. To remove bias from GNN predictions, these methods design different approaches including dropping edges that have been influenced by bias [68], modifications on the graph structure or node features, and node [292]. For link prediction tasks, Li et al. [34] propose a method called FairAdj and dyadic fairness, which is defined as follows:

\[
\Pr(f(u, v) \mid S(u) = S(v)) = \Pr(f(u, v) \mid S(u) \neq S(v))
\]

where \( f(u, v) \) is the link prediction function for node \( u \) and node \( v \), and \( S(\cdot) \) indicates the sensitive attribute value of nodes. FairAdj derives the bound for the discrepancy in link prediction expectations between intergroup and intragroup nodes. To minimize the upper bound, FairAdj is formulated as a bilevel optimization that iteratively optimizes the parameters of GNNs and fair connection to achieve fair link prediction. Another instance of fair graph structure methods is the method called FairDrop [68], which utilizes edge-drop operations to reduce the unfairness impact caused by a original graph structure.

Regularization methods add extra regularization terms to the objective functions or final predictions of GNNs to improve fairness. For example, a method called InFoRM [35] utilizes the similarity matrix defined on nodes to obtain the regularization term, which measures the individual fairness of GNNs. Notably, as nodes in a graph are connected by edges, it is not easy to achieve individual fairness on nodes by considering node features alone. InFoRM [35] employs a variant of the \((d_1, d_2)\)-Lipschitz property to measure the individual fairness of nodes. It is defined as

\[
\sum_{i=1}^{n} \sum_{j=1}^{n} \|Y[i, :] - Y[j, :]\|_2^2 S[i,j] = 2 \text{Tr}(Y'L_S Y) \leq \delta
\]

where \( Y[i, :] \) and \( Y[j, :] \) are the graph mining results of node \( i \) and \( j \), \( S[i,j] \) indicates the similarity of nodes, and \( \delta > 0 \) is a constant. \( L_S \) represents the Laplacian matrix of \( S \). \( \text{Bias}(Y, S) = \text{Tr}(Y'L_S Y) \) is used to measure the overall bias in the result \( Y \) with respect to similarity \( S \). To achieve debiasing of GNNs, the item \( \text{Bias}(Y, S) \) is added to the loss function as a fairness regularization item during the training of GNNs. Moreover, this regularization item can also be employed in the debiasing of input graph data and graph mining results. Thus, InFoRM can flexibly boost GNN fairness in any processing phase (preprocessing, in-processing, or postprocessing). Another typical example is the REDRESS [284] method, which defines a regularization term from the ranking perspective.

Remarks: In the Appendix, we present further related resources for studying the fairness of GNNs. For fairness in general graph mining, the tutorial [283] introduces the definition of fairness in various tasks and the typical methods used to achieve it.

C. Summary

Table 6 presents a comparison of the above-mentioned methods for improving the fairness of GNNs. Existing fairness enhancement methods on GNNs can be categorized into preprocessing, in-processing, and postprocessing methods. Preprocessing methods generally require modification of the training graph dataset (e.g., InFoRM-G [35]) of GNNs to remove potential discrimination in the data. In-processing methods modify the operation pipeline (e.g., learning fair graph structure in FairAdj [34] and introducing an adversary module in FairGNN [23]) or loss function (e.g., REDRESS [284]) to improve fairness in GNN predictions. However, both preprocessing and in-processing methods are practical only when users are permitted to modify graph data and GNN models, respectively. If users can only treat GNNs as black boxes (e.g., when using cloud-based GNN services [190]), then users can...
only query GNNs and employ postprocessing methods (e.g., InFoRM-R [35]) to alleviate predication unfairness in target GNNs.

When alleviating the bias in GNN systems, different fairness methods face different degrees of deployment complexity. First, when deployed in the same scenario, different methods have different time complexities. For example, when improving the group fairness of link prediction tasks, a method called FairDrop [68] can directly modify the graph structure without learning. However, FairAdj [34] requires an additional training process to modify the graph structure, where gradient backpropagation and sort-based projections are necessary. Second, the choice of fairness methodology also affects space complexity. For example, when an adversary module (e.g., FairGNN [23]) is introduced to improve fairness by modifying GNN parameters, additional space is needed to deploy this module and train it. In contrast, without requiring additional space, regularization methods (e.g., InFoRM [35]) can generally be added to the loss function in a plug-and-play manner. Finally, it is necessary to take into account the impacts on other aspects (e.g., privacy) of trustworthy GNNs when implementing fairness methods. Refer to Section IX for more details.

D. Applications

When serving various types of users, GNN systems must eliminate bias to deliver high-quality and fair services. Next, we show how common applications require fairness from GNN systems.

1) Social Networks: To provide better service to users, social network analysis has been widely conducted on social media platforms (e.g., Twitter). Considering that online social networks majorly evolve as connections between users change, it is important to ensure fairness when pushing recommendations to users [270], which avoids their complaints about biased user recommendations. Moreover, when GNNs are employed in node classification (e.g., user profile completion [293]) by social media for providing better services, GNN predictions (e.g., occupations and income level) should not depend on sensitive user attributes (e.g., region, nationality, or age) [23].

2) Recommender Systems: As data-centric platforms, such as recommender systems [23], [270], [271], [272], become increasingly popular [294], there is concern about potential bias regarding users and items, who are two typical stakeholders in recommender systems [295]. The fairness concerning users aims at mitigating any discriminatory recommendation practices against certain users (i.e., individual fairness [296], [297]) or demographic groups (i.e., group fairness [292]); the fairness concerning items seeks to ensure equitable visibility for various items, eliminating any favoritism toward more popular options [298], [299]. The recommender systems of today strive for various forms of fairness [299], such as multi-sided fairness for multiple roles [300], interaction fairness [301], causal fairness benefiting from reasoning [302], and dynamic fairness against time bias [303].

Remarks: Note that bias toward user groups with protected attributes (e.g., gender) should also be avoided [67] in other tasks such as financial analysis (e.g., loan default risk prediction [304], [305] and fraud detection [306]), recidivism predictions, and salary prediction [23]. Furthermore, according to the unfairness cycle in Fig. 4, the bias in data also contributes to unfair predictions, which highlights the need for fairness-aware data collection and preprocessing.

E. Future Directions of GNN Fairness

1) Fairness Definition and Evaluation: The fairness of GNNs is an emerging topic in GNN research. The definition of fairness varies from method to method and is influenced by the task at hand. Some methods emphasize group or individual fairness, while others focus on improving the independence between the learned representations and sensitive attributes [23] or the robustness of GNNs to the perturbation of sensitive attributes [67]. Thus, it is essential to arrive at a reasonable definition of fairness for various kinds of graph tasks (e.g., exploring dynamic graphs [307] or heterophilic graphs [308]) and applications, which is essential for building fairness in trustworthy GNNs. Moreover, the diverse range of definitions for the concept of fairness results in different metrics being employed for fairness evaluation. It is also necessary to build unified evaluation metrics and datasets.

2) Influence on Task Performance: The influence of promoting fairness on the original task performance of GNNs is unclear owing to complexity of graph data and the wide range of fairness definitions. On the one hand, introducing fairness to GNNs reduces GNNs’ dependence on sensitive attributes. Existing research into algorithmic fairness [309] demonstrates that information reduction is harmful to model performance, resulting in the performance degradation of GNNs when considering fairness. On the other hand, a novel view of fairness may boost the performance of GNNs in some applications. For example, RawlsGCN [310] introduces the Rawlsian difference principle to GCN to alleviate the degree-related unfairness; results show that it can improve the performance of GCN on some datasets. To build trustworthy GNNs, it is necessary to explore the correlation between the fairness and performance of GNNs.

3) Revealing Unfairness: Although existing studies have demonstrated the unfairness of GNNs and proposed various methods to promote fairness, they cannot provide detailed explanations for the discrimination detected in predictions. One promising direction for fairness research is that of revealing the sources of unfairness [311], [312], [313], [314], as these findings can subsequently guide the design of GNN architectures concerning fairness. For example, a method called FairAdj [34] has revealed that...
unfair connections within the graph structure may cause bias in predictions and learn a fair graph structure to promote group fairness. Moreover, exploring the relations between the explainability and fairness of GNNs is also helpful for providing insights that may help to alleviate discrimination in the GNN context.

VII. ACCOUNTABILITY OF GNNs

Accountability refers to the extent to which people can trust GNNs by assessing a complex GNN system. In practice, GNN systems are used as black boxes that provide little transparency to either developers or users. For example, cloud-based GNN services perform their functions in the cloud [190]; their underlying operations cannot be directly accessed by end users. In the absence of proper accountability, users will not fully trust GNN service providers since they have concerns due to frequent security incidents on the cloud [315]. In addition, there are large numbers of blocks in today’s graph-based GNN systems and various roles in play when developing and maintaining such complex systems. It is necessary to identify who or what is responsible for each component when an error occurs. All the above concerns give rise to the requirement for accountability.

Due to the diverse range of application scenarios and research focuses, different researchers have proposed different definitions of accountability. In general, the concept of accountability encompasses a number of aspects [316], including the following.

1) Detection: A system is accountable if it can provide a means to detect violations or misbehavior.
2) Evidence: Accountability refers to the ability to present evidence that might convince a third party.
3) Identification and association: Accountability is the property through which the state and actions of identities can be associated in a way that enables any misbehavior to be detectable, provable, and undeniable.
4) Answerability: Accountability is the notion of being liable to answer for conduct (e.g., incident management and remediation).
5) Blame: Accountability refers to the ability to assign responsibility for actions based on additional analysis beyond the direct mapping of identification.
6) Punishment: A system is accountable if it can impose sanctions in the event that responsibilities are not met.

In light of the above definitions and the guidelines provided by the European Commission [317], we outline three common objectives that should be referenced when building an accountable GNN: 1) design a set of formable assessment and certification processes throughout the entire development cycle of GNN systems that enable violations or misbehavior to be detected and associated with specific identities (detection, identification and association, and blame); 2) provide assurance of auditability, e.g., by providing open access to event logs in order to provide a record that will be convincing to third parties (evidence); and 3) set up explainable coordinations such that humans will be capable of adjusting, remediating, and punishing the actions of a GNN with ease (answerability and punishment).

From a technical perspective, the majority of existing studies concerning GNN accountability focus primarily on achieving the first objective, namely, proposing formable assessments of GNNs to facilitate the detection of violations. In this section, we will present some of these methods and then go on to propose potential directions for research into enhancing accountability.

A. Methods

According to the type of violation involved, existing technologies for improving GNN accountability can be divided into two categories: those that detect violations of utility and those that detect violations of security. A violation of utility indicates that the GNN is not performing as expected (particularly with regard to its accuracy). A violation of this kind can be detected and analyzed via benchmarking frameworks, which can be used to assess the GNN performance (i.e., to determine whether it is performing as expected) and to identify the corresponding impact factors (associating the violation with an identity). Moreover, a violation of security refers to a breach of security policies [82] (confidentiality, integrity, and availability) and can be detected by security evaluation techniques. Note that there may be other methods for detecting these two types of violations, as well as other violations in the GNN context (e.g., violations of the law or ethics).

1) Benchmarking: To comprehensively assess GNNs over their entire life cycle, it is necessary to propose standardized evaluation methods for every step of GNN development. Benchmarking frameworks can be created to determine whether GNN models have been properly constructed, trained, and validated. Categorized by the different stages of GNN development, they are listed as follows.

a) Architecture design: Due to the wide variety of different applications and graph types, there are numerous kinds of GNN models and architectures. Thus, several research projects attempt to develop benchmarking frameworks to assist developers in selecting the appropriate model hyperparameters [36], [318].

b) Model training: Due to the lack of transparency in model training, a set of works [319], [320] have attempted to examine how various training strategies (e.g., training hyperparameters and optimization problem constructions) can impact the performance of GNNs.

c) Model validation: Following model training, it is necessary to evaluate the final performance of GNNs. A set of studies [36], [321] have sought to develop comprehensive frameworks for evaluating
Zhang et al.: Trustworthy GNNs: Aspects, Methods, and Trends

GNN models, which can assist users and developers in validating their GNNs.

2) Security Evaluation: To evaluate the security properties of a GNN (i.e., confidentiality, integrity, and availability), several security evaluation techniques can be applied. While these techniques have previously been employed in general-purpose computing systems, most existing studies on ML-based systems have focused on integrity verification. Due to the similarities between the development and implementation processes of GNNs and DNNs, these methods can be directly applied or easily adapted to GNNs despite having been developed for DNNs. Depending upon the objective of the verification, there are two types of integrity verification: data integrity verification and procedure integrity verification.

a) Data integrity: Data (including graphs and models) are one of the key components of GNN systems. Most current studies focus on verifying the integrity of models during inference. Javaheripi and Koushanfar [322] propose to verify the integrity during running time by means of a unique hash signature extracted from the benign model prior to its deployment. For their part, He et al. [323] construct sensitive-sample fingerprints whose outputs can clearly reflect even small changes to the model.

b) Procedure integrity: Aside from data in GNNs, it is also imperative to ensure the integrity of each step in the system development process. Most current works focus on verifying the training procedure [324], [325]. For example, Jia et al. [324] combine studies in proof of work and verified computations to verify the training process by analyzing the initial models, intermediate models, and final trained models.

Remarks: Note that the above verification methods introduce additional costs either before the deployment or during the verification. For example, verifying model integrity via fingerprints requires pregenerated and selected sensitive samples before the model deployment [323]. Verifying the training requires recording specific manufacture during the training, whose complexity is affected by the checkpointing interval and the size of the training dataset [324].

B. Applications

While limited studies focus on constructing accountable GNNs, the need for incorporating accountability in building trustworthy GNNs is crucial. In this section, we highlight applications where GNNs play a pivotal role and where accountability emerges as an essential consideration.

1) Crime Prediction: Recent research indicates that GNNs can capture interregional crime patterns and forecast criminal activities [326], [327]. Given the sensitive nature of this domain, concerns about the GNN’s accountability naturally arise. Specifically, whether the inference process is transparent and well-referenced is especially pertinent due to policy implications in this application [328].

2) Chip Design: GNNs have also made inroads into chip design. A notable application is their use in detecting HW Trojans in chips sourced from untrusted foundries [329]. Here, GNNs analyzing integrated circuit layouts demand pinpointing suspicious regions and the provision of substantial evidence. These match the requirements of accountability and underscore the importance of GNNs being accountable.

C. Future Directions of GNN Accountability

Despite the urgent demands to address accountability in GNN systems, there have been few attempts to do so. On this subject, several future research directions should be considered.

1) Detection and association of other types of violations. Current research into building accountable GNNs focuses only on detecting violations of utility and security. However, there are other factors that should be taken into consideration, such as ethical, legal, and environmental violations. It is also necessary to conduct consistent assessments of these violations and evaluate their effects.

2) Verification of other components of GNN systems. The verification techniques introduced above tend to focus on either the training process or trained GNN models. Nevertheless, the security of other components (e.g., the integrity of graph data and the inference process of GNN models) should also be guaranteed.

3) Development of mechanisms to achieve other objectives (e.g., providing auditability and explainable coordination) of building accountable GNNs.

VIII. ENVIRONMENTAL WELL-BEING OF GNNs

Competent GNNs also should conform to the fundamental values of the society in which they are deployed. A shared expectation of societies with diverse cultural backgrounds is that GNNs should perform competently and efficiently across their range of applications. One study on energy consumption [330] shows that training a common NLP model produces the same amount of carbon dioxide as a human would produce in seven years. To reduce the environmental impacts of GNNs and ensure their trustworthiness, it is crucial to explore the environmental well-being of GNNs.

In this survey, our account of the environmental well-being of GNNs focuses on methods that can improve GNN efficiency when they are faced with various practical challenges. Some representative efficiency-related challenges are given as follows.

1) Given the explosive growth of graph data, large-scale datasets [331] pose a significant challenge to efficient GNN execution in terms of both training and inference.
2) Deeper or more complex GNN architectures [20, [332], [333] have been proposed to obtain better task performance, overcome oversmoothing, or reduce overfitting [334]. However, the sizes of these models challenge their deployment on edge devices (e.g., autonomous vehicles [75]) with limited computational resources (e.g., memory) [335].

3) Due to the unique characteristics of graph data (e.g., irregularity), the study and deployment of GNNs would benefit significantly from specially designed and efficient SW frameworks and HW accelerators [73].

In this section, we review recent methods for improving the efficiency of GNNs, which is regarded as aligning GNN research with social values regarding environmental well-being. Generally, the efficiency improvement is evaluated with reference to time-related metrics (e.g., response latency or speedup rating [336], [337], throughput rating [338], [339], and communication time [340]), energy-related metrics (e.g., nodes-per-joule [38] and energy consumption [341]), or resource-related metrics (e.g., memory footprint [76], cache access performance, and peak memory usage [342]). Existing methods include scalable GNN architectures and efficient data communication, model compression methods, efficient frameworks, and accelerators.

A. Methods

1) Scalable GNN Architectures and Efficient Data Communication: Scalable GNN architectures are GNNs that can efficiently process graphs on a giant scale. Recently, giant-scale graphs obtained from real-world scenarios have made the application of GNNs more challenging. For example, even an online gaming graph called Friendster might contain 65.6 million nodes and over 1.8 billion edges [343]. To handle large-scale graphs, some studies have explored sampling methods [50], [344], scalable architectures [20], [51], and industrial applications with GNNs [5], [345]. Moreover, some GNNs suffer from inefficient data loading. For example, data loading occupies 74% of the whole training time for GCN [339]. A method called PaGraph [339] analyses pipeline bottlenecks of GNNs and proposes a GPU cache policy to reduce the time consumption associated with moving data from CPU to GPU.

2) Model Compression Methods: Due to the irregularity of graph data, the exponential expansion of computational graphs challenges the efficiency of GNNs when stacking GNN layers, which limits the applicability of deep GNNs in real-time systems. To balance the efficiency and competence of GNNs, recent studies have introduced knowledge distillation, model pruning, reducing parameters, and model quantization to GNNs for compressing GNN models.

Knowledge distillation aims to utilize competent deep GNNs to train lightweight GNNs that can achieve similar performance [346], [347]. For example, a method called TinyGNN [336] is proposed to learn local structure knowledge from a GNN model by distilling neighbor nodes. Another effective approach for acceleration is model pruning. One representative method, called UGS [348], introduces the lottery ticket hypothesis to GNNs, which can prune and accelerate GNNs on large-scale graphs. Moreover, reducing the size of GNN parameters can also improve their efficiency. For example, a method called GNF [76] employs normalizing flows to achieve reversible GNNs, which reduces the memory footprint of GNNs and improves their practicability. In addition, the inference of GNNs can be accelerated with the help of low-precision integer arithmetic (i.e., model quantization) [349]. Recently, an architecture-agnostic method called Degree-Quant [337] has demonstrated how to implement quantization-aware training, which can achieve efficient and competent GNNs.

3) Efficient Frameworks and Accelerators: The computing of GNNs presents a series of efficiency challenges, including computational dependence on input graphs, intertwined computation between dense and sparse operations, and a diversity of applications and tasks [73]. To this end, several SW frameworks and HW accelerators have been proposed to facilitate the development of efficient GNNs [73].

Different graph SW frameworks have been developed to provide efficient interfaces for GNN implementations. These frameworks are designed for sparse operations or efficient execution on various graphs (e.g., spatiotemporal graphs) in a multi-GPU environment. Typical SW frameworks [73] include PyTorch Geometric (PyG) [350], deep graph library (DGL) [351], and so on. In addition to the above SW frameworks, HW accelerators have emerged to streamline the execution of GNNs. These HW accelerators are energy-efficient and increase the operating speed of GNNs by two to three times [73]. Typical HW accelerators include the EnGN [352], HyGCN [341], and so on. In addition, some studies have focused on analyzing the efficiency bottleneck of GNNs [353] and an SW-HW co-design for the accelerator [354].

Remarks: Generally, technologies for the environmental well-being of GNNs can extremely reduce the complexity of deploying GNN systems. However, it is not trivial to balance the tradeoff between the performance and efficiency of GNNs due to the size deduction of training data (e.g., sampling [344]) or model parameters (e.g., model distillation [346]). Moreover, due to the diversity of graph data and GNN architectures, it is challenging for the design of SW frameworks and HW accelerators to balance their generality and specificity. Note that improving other aspects (e.g., robustness) of trustworthy GNNs is generally at the cost of environmental well-being (refer to Section IX for more details).

B. Applications

Environmental well-being (i.e., efficiency) facilitates the technology ecosystem of GNN systems from diversified
aspects. Next, we introduce typical applications of the environmental well-being of GNNs.

1) Algorithm Design: During the training phase, for giant scale graphs (e.g., the user-item graph on Pinterest with data size 18 TB [355]), sampling methods and scalable algorithms are desired to improve training efficiency [356], [357], while, in the inference phase, users expect smaller and faster new models that can maintain similar performance (e.g., accuracy) as the well-trained GNNs. The efficiency needs can be satisfied by using model compression methods [357].

2) HW Accelerators: In addition to GNN algorithms, the environmental well-being of GNNs can also be applied in devising customized HW accelerators of GNNs [358]. Due to the diversity of graph data (e.g., dynamic graphs), various GNNs have been proposed to explore graph data for better performance. However, traditional accelerators designed with the consideration of computation generality potentially prevent them from obtaining optimal efficiency performance, raising the demands on customized graph HW [341].

C. Future Directions of Environmental Well-Being in GNNs

1) Exploration of Efficient GNNs: Although improving the efficiency of AI systems has attracted the attention of AI practitioners, the exploration of efficient GNNs has been limited. For various types of graph data (e.g., spatiotemporal graph) and applications, one direction is to analyze the efficiency bottleneck of current GNNs and then improve them or explore new and more efficient operations in GNNs. In GNN deployment, accelerating the inference of GNNs [338], [342] is a practical approach for achieving more efficient GNNs. Compressing models and accelerating the inference of GNNs will directly contribute to the environmental well-being of GNNs. Moreover, introducing efficiency considerations into GNN training [337] will also promote the development of efficient GNNs. Adaptive architecture designs aimed at saving energy [45], [359] have been proposed in the context of CNN models; thus, it would also be worthwhile to analyze the energy consumption of different operations in GNNs and their operational complexity since these factors may contribute to efficient GNN architectures by Graph NAS [360].

2) Accelerators for GNNs: Current research into accelerators for GNNs focuses on designing SW frameworks and HW separately. We argue that the acceleration of GNNs should take the whole pipeline of GNNs—consisting of the modeling, applications, complexity, algorithms, accelerators, HW/SW requirements, and data flow—into account. By identifying the actual requirements of GNNs in applications (e.g., quick response in real-time systems and limited memory footprint in edge computing terminals) and the bottlenecks in the execution computation of GNNs (e.g., data flow between CPU and GPU), the SW-HW codesign [361] is a practical and promising direction for future research aimed at achieving the environmental well-being of GNNs.

IX. RELATIONS BETWEEN ASPECTS OF TRUSTWORTHY GNNs

Most studies in Sections III–VIII focus on one specific aspect of trustworthy GNNs. However, when building trustworthy GNN systems, it is necessary to take complicated cross-aspect relationships into account; this is because, in practical development and deployment, the above six aspects of GNNs are highly related, and an interplay (e.g., accordance and tradeoff) exists among them. To thoroughly understand and build trustworthy GNN systems, it is both inevitable and imperative to explore and study the complex relations between these aspects.

In this section, we present existing studies of these relations (as shown in Table 7) by with focuses on the following: 1) how the methods from one aspect of trustworthy GNNs are adapted to address objectives in other aspects and 2) why advancing one aspect of trustworthy GNNs can promote or inhibit other aspects.

A. Explainability and Robustness

The methods developed to explore GNN explainability can be used to implement adversarial attacks and defenses on GNNs. Explanation methods reveal the behaviors of GNNs by identifying why a GNN makes a specific prediction with regard to the current samples. Thus, they provide insights that support the design of attack and defense algorithms. For example, in explainability-based backdoor attacks [102], GNNExplainer is employed [24] to identify the importance of nodes and guide the selection of the injected backdoor trigger position. Moreover, defenders can also use explainers for GNNs to identify the potential locations of malicious edge perturbations [149]. By introducing information bottleneck into GNNs, another method called GIB [166] owns better performance on noisy graph classification.

B. Robustness, Fairness, and Privacy

Methods for improving the robustness of GNNs can be adapted to aid in improving GNN fairness. For example, adversarial training is a typical method for achieving robust GNNs by perturbing training data. Adopting a related perspective, a method called NIFTY [67] follows a strategy similar to adversarial training to perturb sensitive attributes in order to counteract the unfair GNN predictions regarding these attributes. The privacy of GNNs also potentially benefits from the improved GNN fairness. For example, fair representation learning methods designed to improve the fairness of GNNs also reduce the risk of private information leakage from sensitive attributes [23], [290], as it is difficult to infer these attributes from the learned representations. However, another recent study shows that increasing the individual fairness of nodes can potentially improve the privacy risks of edges [365].
Table 7 Representative Methods for Trustworthy GNNs

| Robustness  | Explainability  | Privacy  | Fairness  | Accountability  | Environmental Well-being | Others  |
|-------------|-----------------|----------|-----------|-----------------|---------------------------|--------|
| [46], [55], [120], [94], [116], [114], [113], [96], [135], [362] | [167], [61], [24], [59], [154], [172], [182], [187], [183], [173] | [216], [63], [250], [363], [244], [253], [47], [364] | [365], [290], [23], [290], [35], [34], [67], [68], [284], [23] | [290] | [369] | [81], [79], [80], [374], [375] |

* The references at diagonal positions indicate representative studies for one aspect of trustworthy GNNs. The references at non-diagonal positions represent recent studies related to complex cross-aspect relations, where methods/advancements from one aspect can affect another aspect of trustworthy GNNs.

\* denotes studies showing how the methods from one aspect of trustworthy GNNs are adapted to address objectives in other aspects, and \* marks the studies that discuss why advancing one aspect of trustworthy GNNs can promote or inhibit other aspects. Moreover, references with/without underline indicate studies that present potential/clear relations; see Section IX for more details.

C. Environmental Well-Being and Others

Compared with efforts to improve other aspects, methods targeting environmental well-being exhibit more diversity, which results in the existence of multiple relations between environmental well-being and other aspects. For example, methods designed to promote the environmental well-being (i.e., model compression methods) of GNNs can also facilitate exploring the explainability of GNNs. A knowledge distillation method called CPF [368] can be used to help build surrogate models that facilitate GNN explanation. Moreover, environmental well-being has complex advancement-level relations with other aspects of trustworthy GNNs. Several examples are presented in the following.

Achieving environmental well-being (i.e., model compression) of GNNs may contribute to the robustness. The quantization of GNNs, a general approach for improving efficiency, also makes them more robust. Generally, quantized deep learning models are more robust to noise and adversarial attacks [376]. The quantization of models can be considered as adding extra terms in objective functions, which reduces overfitting risks and increases the generalization ability and robustness of models. Recent research shows that such insights can be also applied to GNNs [366].

Better accountability techniques also improve the environmental well-being of GNNs. Specifically, accountability in the GNN context involves implementing a series of processes to assess the GNN systems. Meanwhile, most of these studies also aim to assess GNN systems efficiently. The improvement of accountability techniques can, thus, also promote a higher level of GNN environmental well-being. For example, Lacombe et al. [369] develop a framework that provides developers with valuable knowledge regarding GNN training. Specifically, it indicates whether the training of GNNs is processing as expected, which is helpful for stopping futile training and reducing energy consumption.

Building robust GNNs can also impact the environmental well-being of GNNs. Graph structure learning is a typical robustness enhancement method for GNNs, but its clean graph learning process (e.g., training an extra graph generation module [114], [367], [377]) increases the development cost of GNNs. Moreover, introducing specially designed operations in GNN architectures [116], [117] improves GNN robustness; however, more
operation efforts are needed to develop and implement these modified GNNs. These robustness-oriented efforts lead to higher energy costs, which are detrimental to the environmental well-being of trustworthy GNNs.

The implementation of privacy techniques for GNNs also increases the costs of developing and implementing GNN systems, which hinders the environmental well-being of trustworthy GNNs. Generally speaking, achieving privacy in the GNN context necessitates the addition of extra processes or specific designs, which, in turn, increases the amount of development effort required. For example, a privacy-preserving GNN architecture reduces the leakage of sensitive attributes but increases GNN training costs [64].

Remarks: Note that it is not trivial to unravel the complex relations between aspects of trustworthy GNNs. The literature on the relations in AI contexts is provided here, which is expected to facilitate further relation study when building trustworthy GNNs. For robustness and explainability, a recent tutorial [378] demonstrates that improving the robustness of neural networks contributes to more visually appealing explanations when using gradient-based attribution methods. In the area of privacy and fairness, a survey [379] outlines five prominent architectures for implementing privacy and fairness simultaneously, followed by summarizing how enhancing one of them can affect the other.

X. CONCLUSION AND TRENDING RESEARCH DIRECTIONS

Trustworthy GNNs is a thriving research field with a wide range of methodologies and applications. This survey summarizes current advancements and trends, which is expected to facilitate and advance future research into, and implementation of, trustworthy GNNs.

From the perspective of technology, this survey presents a roadmap toward comprehensively building trustworthy GNNs. We first define trustworthy GNNs and depict their characteristics by categorizing principles in the existing literature related to trustworthiness. Next, we review recent advancements in trustworthy GNNs from the aspects of robustness, explainability, privacy, fairness, accountability, and environmental well-being. For each aspect, we introduce the basic concepts, comprehensively summarize the existing methods for building trustworthy GNNs, and then point out potential future research directions related to these aspects. In addition, we emphasize that cross-aspect study is vital for achieving trustworthy GNNs. We present complex relations between aspects by introducing methodology transfer and discussing the points of agreement and conflict between each aspect.

While trustworthy GNNs have attracted increasing attention, applying them to real-world applications still poses many challenges; these range from the design philosophy of GNNs to the conceptual understanding of trustworthiness and from the diversified relations between aspects of trustworthy GNNs to the practicability of the proposed methods in specific real-world applications. Moving away from the future directions suggested for specified aspects of trustworthy GNNs in Sections III–VIII, we here highlight six trending directions with the potential to fill the research gap and boost the industrialization of trustworthy GNNs by considering the concept as a whole.

A. Shift to Trustworthy GNNs

Building trustworthy GNNs requires researchers to embrace the trustworthiness-related concepts and characteristics introduced in this survey and shift their focus from performance-oriented GNNs to trustworthy GNNs. With regard to achieving this shift, introducing trustworthiness into the design philosophy of GNNs is a challenging yet promising direction. An example of incorporating explainability is a method called GIB [166], which can simultaneously recognize an information bottleneck subgraph (as an explanation) for the input graph and cooperate with other GNN backbones in graph classification. As an example of incorporating fairness, RawlsGCN [310] alleviates the degree-related unfairness by introducing the Rawlsian difference principle [380] (i.e., maximizing the welfare of the worst-off groups [310]) to GCN. Moreover, some open problems need to be solved if trustworthy GNN systems are to be realized, for example, that of how to balance the tradeoff between certain aspects (e.g., robustness and environmental well-being) in specified applications (e.g., autonomous vehicles [75]). Thus, shifting to trustworthy GNNs and solving these problems are fundamental for building trustworthy GNN systems.

B. Other Aspects of Trustworthy GNNs

The content of trustworthy GNNs goes beyond the six aspects discussed in this survey. For example, generalization ability is considered to be a necessary aspect of trustworthy systems [11]. Current research into the extrapolation of GNNs [81] aims to reveal the relationship between tasks and aggregation functions, which will promote further study of the generalization of GNNs and guide the design of trustworthy GNNs. Therefore, studying the proper handling and management of the trustworthiness-related principles outlined in Table 1 (e.g., safety and controllability, and respect for autonomy) is another vital research direction in building human-centered and trustworthy GNN systems. The open framework proposed in this survey (see Fig. 1) will hopefully be able to incorporate aspects of trustworthiness that emerge in the future, thereby expanding the domain of trustworthy GNNs and enhancing the practical implementation of trustworthy GNN systems.

C. Diversified Relations

This survey touches on only a small fraction of the intricate cross-aspect relations. It will, therefore, be crucial to investigate other cross-aspect relations, such as
explainability and privacy, in future work. Moreover, these cross-aspect relations are not only intricate but also exist at multiple levels. For example, counterfactual fairness is conceptually similar to robustness, as both of them expect GNNs to make unchanged predictions when some attributes of the same sample are changed. This clearly suggests that methods for improving GNN robustness can also be used to improve the fairness of GNNs to a certain extent (e.g., by introducing adversarial training on GNNs with respect to sensitive attributes [67]). Unlike traditional studies, a recent study [364] extends the focus of fairness from GNN performance to GNN privacy. After a theoretical analysis of the uneven vulnerability of edges, a group-based attack is proposed to reveal the practical privacy risks of edges against link stealing attacks [364]. Hence, exploring the diversified relations at different levels (e.g., concepts, methods, and advancements) is also essential to comprehensively understanding and practically building trustworthy GNNs.

D. Model-Agnostic Methods

Many methods for trustworthy GNNs require custom-designed network architectures, which makes them unable to improve the trustworthiness of the target GNNs without modifying their architectures. This weakness severely limits the practicability of these trust-enhancing methods for GNNs. By contrast, model-agnostic methods can increase the trustworthiness of GNNs simply by being plugged in to any phase of the system pipeline (i.e., preprocessing, in-processing, and postprocessing). Another advantage of these methods is that they can be combined to serve as building blocks for trustworthy GNNs. Thus, the design of model-agnostic methods is a promising direction for trustworthy GNNs.

E. Technology Ecosystem for Trustworthy GNNs

As an emerging field, research into trustworthy GNNs is inseparable from the surrounding technological ecosystem, including the tools, datasets, metrics, pipelines, and so on. Current tools (e.g., AI 360 tools from IBM [381]) are mainly used to improve the credibility of general machine learning models. However, it may not be easy to apply these tools directly to GNNs owing to the unique characteristics of graphs. For example, updating the embedding of nodes in GNNs usually relies on the edges between nodes. The presence of these edges causes the relationship between nodes to break the independent and identically distributed assumption in general machine learning models, which constitutes a challenge for studying the individual fairness of GNNs. Moreover, various datasets [331], [382], metrics [383], standardized evaluation pipelines [36], [318], and SW platforms [384], [385] that are suitable for miscellaneous applications will also be required if efforts to develop trustworthy GNNs are to be successfully evaluated. Therefore, developing a technological ecosystem that supports trustworthiness is an essential step in researching and industrializing trustworthy GNNs.

F. Opportunities in the Era of New Technologies

As technology evolves endlessly, transformer-based approaches [386] have emerged in the AI community and advanced current graph learning methods. However, trustworthiness issues still exist in systems driven by graph transformers [387]. In addition to the well-known concerns about quadratic running time with respect to number of nodes [388], users may not trust graph transformers due to the absence of explainability in these models [389]. Existing studies demonstrate that transformer-based methods are still vulnerable to adversarial attacks [390], [391]. The above issues suggest opportunities for devising trustworthy GNN learning methods in the new technology era. Note that, due to the fact that both graph data and learning methods contribute to graph learning systems, the trustworthiness (e.g., unfairness) of these systems can also be attributed to the training data itself (e.g., biased graph structure) [35], [43], which highlights opportunities in graph data management (e.g., data-centric graph learning [392], [393]).

APPENDIX

DATA SETS FOR TRUSTWORTHY GNNs

Although some aspects of trustworthy GNNs (e.g., robustness) have been continually explored in recent years, the study of other aspects remains in the initial stages. Current studies on robustness, privacy, accountability, and environmental well-being can be implemented by evaluations on common graph datasets (e.g., TUDataset [382] and OGB [394]); however, when studying the explainability and fairness of GNNs, general graph datasets are not suitable because they do not contain ground-truth explanations and sensitive attributes, respectively. In this section, we briefly review datasets for evaluating the explainability and fairness of GNNs.

1) Explainability: The biggest challenge associated with evaluating explanation methods for GNNs is the lack of ground-truth datasets. Due to the complexity of graph data in real-world applications, it is difficult for humans to understand which components of graph data are the ground-truth reasons for the predictions made by GNNs. Accordingly, some methods use synthetic data to evaluate the quality of explanations. In these datasets, different graph motifs are attached to the base graph. To facilitate the easy evaluation of these explanations, the node labels and graph labels are defined by their role in the synthetic dataset and the motif type that they contain, respectively. The most widely used datasets of this kind are BA-shapes, BA-Community, Tree-Cycle, Tree-Grids, BA-2Motifs [31], and Syn-Cora [163].

Although it is human-intelligible to employ synthetic datasets in evaluation, these synthetic datasets cannot provide adequate benchmarks for complex and various
2) Fairness: Current methods use various datasets to evaluate the fairness of GNNs. For similarity-based fairness (e.g., individual fairness), datasets for node classification and link prediction are used to conduct fairness evaluation [284]. For fairness concerning protected or sensitive attributes (e.g., group fairness), these methods choose datasets that contain sensitive or protected information, such as that pertaining to nationality or region [23]. Some typical datasets are Freebase 15k-237, MovieLens-1M, Reddit [290], Pokec, NBA [23], German Credit, Recidivism, Credit Default [67], Oklahoma97, UNC28 [35], Dutch School, Facebook, and Google+ [395].
B. Wang, J. Guo, A. Li, Y. Chen, and H. Li, "Strengthening linkability in graph neural networks," in Proc. USENIX Secur., 2021, pp. 2669–2686.

X. He, R. Wen, Y. Wu, M. Backes, Y. Shen, and Y. Zhang, "Node-level membership inference attacks against graph neural networks," 2021.

B. Wu, Y. Yang, S. Pan, and X. Yuan, "Adapting membership inference attacks to GNN for graph classification: Approaches and implications," in Proc. IEEE Int. Conf. Data Mining (ICDM), Dec. 2021, pp. 1421–1426.

H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "GraphSAINT: Graph sampling based node embeddings," in Proc. Int. Conf. Learn. Represent., 2020.

B. Wu, X. Yang, S. Pan, and X. Yuan, "Adapting membership inference attacks to GNN for graph classification: Approaches and implications," in Proc. Int. Conf. Data Mining (ICDM), Dec. 2021, pp. 1421–1426.

P. Doshi-Velez et al., "Accountability of AI under the law: The role of explanation," 2017, arXiv:1711.01134.

J. A. Kroll, "Accountable algorithms," Ph.D. dissertation, Dept. Comput. Sci., Princeton Univ., Princeton, NJ, USA, 2015.

Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," IEEE Trans. Knowl. Data Eng., vol. 34, no. 1, pp. 249–270, Jan. 2022.

S. Abadal, A. Jain, R. Guirado, J. López-Alonso, and E. Alarcón, "Computing graph neural networks: A survey from algorithms to accelerators," ACM Comput. Surv., vol. 54, no. 9, pp. 1–38, Dec. 2020.

B. Wu, X. Yang, S. Pan, and X. Yuan, "Adapting membership inference attacks to GNN for graph classification: Approaches and implications," in Proc. Int. Conf. Learn. Represent., 2020.

W. Zhang and E. Ntousi, "P4NT: An adaptive fairness-aware decision tree classifier," in Proc. 28th Int. Joint Conf. Artif. Intell., Aug. 2019, pp. 1480–1486.

N. A. A. Alif et al., "Graph neural network: A comprehensive review on non-Euclidean space," IEEE Access, vol. 9, pp. 65058–66066, 2021.

Y. Ma and J. Tang, Deep Learning on Graphs. Cambridge, U.K.: Cambridge Univ. Press, 2021.

H. Dai et al., "Adversarial attack on graph structured data," Proc. Int. Conf. Mach. Learn., Stockholm, Sweden, Jul. 2018, pp. 1122–1132.

A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in Proc. Int. Conf. Learn. Represent., 2019.

P. Dey and S. Medya, "Manipulating node similarity measures in networks," in Proc. Int. Conf. Auto. Agents MultiAgent Syst., 2020, pp. 321–329.

D. Luo et al., "Parameterized extractor for graph neural network," in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 1–12.

B. Sánchez-Lengeling et al., "Evaluating attribution for graph neural networks," in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 1–9.

S. R. Popel, S. P. Roostami, C. E. Martin, and H. Hoffmann, "Explainability methods for graph convolutional neural networks," in Proc. IEEE/CVF Conf. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 7067–7073.

J. Ordonez, "The Graph2Class challenge: Adversarial training methods," in Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining, Aug. 2021, pp. 2641–2677.

J. Su et al., "Project attack: A transferable evasion attack method on graph neural networks," in Proc. 30th ACM Int. Conf. Inf. Knowl. Manage., Oct. 2021, pp. 3617–3621.

Q. Zhou, L. Li, N. Cao, L. Ying, and H. Tong, "MDMERING: Adversarial multi-network embedding," in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2019, pp. 1522–1527.

K. Zhou et al., "Structural attack against graph based Android malware detection," in Proc. CCS, Nov. 2021, pp. 3218–3235.

S. Freitas, D. Yang, S. Kumar, H. Tong, and D. H. Chau, "Graph vulnerability and robustness: A survey," 2021.

Y. Xia, J. Fun, and D. Hill, "Cascading failure in Watts-Strogatz small-world networks," Phys. A, Stat. Mech. Appl., vol. 389, no. 6, pp. 1281–1285, Mar. 2010.

D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in Proc. Int. Conf. Learn. Represent., 2019, pp. 1–8.

V. Gupta and T. Chakraborty, "VIKING: Adversarial attack on network embeddings via supervised network poisoning," in Proc. Pacific-Asia Conf. Knowledge Discovery Data Mining in Lecture Notes in Computer Science, vol. 12714, Cham, Switzerland: Springer, 2021, pp. 103–115.

Y. Sun, S. Wang, X. Tang, T. Y. Hsieh, and V. Honavar, "Adversarial attacks on graph neural networks via node injection: A hierarchical reinforcement learning approach," in Proc. Web Conf., Apr. 2020, pp. 673–683.

X. Lin et al., "Exploratory adversarial attacks on graph neural networks," in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2020, pp. 1136–1141.

S. Yu, J. Zheng, Y. Wang, J. Chen, Q. Xuan, and Q. Zhang, "Network embedding attack: An Euclidean distance based framework," in Proc. MDATA: A New Knowledge Representation Model, in Lecture Notes in Computer Science, vol. 12647, Cham, Switzerland: Springer, 2021, pp. 131–151.

Z. Zhang, J. Ju, B. Wang, and N. Z. Gong, "Backdoor attacks to graph neural networks," in Proc. 26th ACM Symp. Access Control Models Technol. (ACMAT), Jun. 2021, pp. 15–26.

J. Wang, M. Luo, F. Suya, J. Li, Z. Wang, and Q. Zhang, "Scalable attack by injecting vicious nodes," Data Min. Knowl. Disc., vol. 34, no. 5, pp. 1363–1389, 2020.

Y. Ma, S. Wang, T. Derr, L. Wu, and J. Tang, "Graph adversarial attack on network embedding based on MDATA: A New Knowledge Representation Model," in Proc. 27th ACM SIGKDD Conf. Knowledge Discovery Data Mining, Singapore, Aug. 2021, pp. 1161–1169.

J. Xu, M. Xue, and S. Picek, "Explainability-based backdoor attacks against graph neural networks," in Proc. 3rd ACM Workshop Wireless Secur. Mach. Learn. (WiseML@WISec), Abu Dhabi, United Arab Emirates, Jul. 2021, pp. 31–36.

X. Zang, Y. Xie, J. Chen, and B. Yuan, "Graph universal adversarial attacks: A few bad actors ruin graph learning models," in Proc. 13th Int. Joint Conf. Artif. Intell., Aug. 2021, pp. 3328–3334.

Y. Chen et al., "Understanding and improving graph injection attack by promoting unnoticeability," in Proc. Int. Conf. Learn. Represent., 2022, pp. 1–8.

J. Chen et al., "MOA: Momentum gradient attack on network," IEEE Trans. Comput. Social Syst., vol. 8, no. 1, pp. 99–109, Feb. 2021.

K. Xu et al., "Topological attack and defense for graph neural networks: A generalization perspective," in Proc. 28th Int. Joint Conf. Artif. Intell., 2019, pp. 3961–3967.

H. Wu, C. Wang, Y. Tsybulevskiy, A. Docherty, K. Lu, and L. Zhu, "Adversarial examples for graph data: Deep insights into attack and defense," in Proc. 28th Int. Joint Conf. Artif. Intell., 2019, pp. 4816–4823.

J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan, "Fast gradient attack on network embedding," 2018, arXiv:1809.02797.
W. Jin et al., “Adversarial attacks and defenses on graph neural networks: A review and empirical studies,” 2020, arXiv:2003.06653.

J. Jin, B. Wang, Q. Li, K. Sun, M. Xu, and Z. Liu, “A hard label black-box adversarial attack against graph neural networks,” 2021, arXiv:2108.09513.

Q. Zheng et al., “Robustness benchmark: Benchmarking the adversarial robustness of graph machine learning,” 2021, arXiv:2111.04314.

N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. Papalexakis, “All you need is low (Rank): Defending against adversarial attacks on graphs,” in Proc. 13th Int. Conf. Web Search Data Mining, Jan. 2020, pp. 169–177.

W. Jin, P. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, “Graph structure learning for robust graph neural networks,” in Proc. 26th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining, Aug. 2020, pp. 66–75.

Z. Xu, S. Du, and H. Tong, “Graph sanitation with application to node classification,” 2021, arXiv:2105.09384.

X. Zhang and M. Zitnik, “GNNguard: Defending against adversarial attacks on graphs,” in Proc. 13th Int. Conf. Web Search Data Mining, Jan. 2020, pp. 600–608.

V. N. Ioannidis, D. Berberidis, and G. B. Giannakis, “Detecting graph convolutional networks against adversarial attacks,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 8469–8473.

Z. Xu, Y. Cao, and Y. Sun, “Towards poisoning of deep learning algorithms with back-gradient optimization,” in Proc. 10th ACM Workshop Artif. Intell. Secur., Nov. 2017, pp. 27–38.

Y. Liu et al., 2022, pp. 6014–6021. A GNN-based imbalanced learning approach for fraud detection,” in Proc. Web Conf., Apr. 2021, pp. 3168–3177.

Y. Dou, Z. Liu, S. Sun, Y. Deng, H. Peng, and P. S. Yu, “Enhancing graph neural network-based fraud detectors against camouflaged fraudsters,” in Proc. 29th ACM Int. Conf. Inform. Knowl. Manage., 2020, pp. 315–324.

A. Visvam and G. Darson, “An efficient Bitcoin fraud detection in social media networks,” in Proc. Int. Conf. Circuit, Power Comput. Technol. (ICCPT), Apr. 2017, pp. 1–4.

J. Pourahmadi, K. -L. Ong, B. H. Kam, and Y. Loo, “Fraud detection: A systematic literature review of graph-based anomaly detection approaches,” Decis. Support Syst., vol. 133, Jun. 2020, Art. no. 113530.

L. Chen, J. Li, Q. Peng, Y. Liu, Z. Zheng, and C. Yang, “Understanding structural vulnerability in graph convolutional networks,” in Proc. 30th Int. Joint Conf. Artif. Intell., Aug. 2021, pp. 2249–2255.

T. Weng et al., “Evaluating the robustness of neural networks: An extreme value theory approach,” in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–12.

F. Yu, Z. Qin, C. Liu, L. Zhan, Y. Wang, and X. Chen, “Interpreting and evaluating neural network robustness,” in Proc. 28th Int. Joint Conf. Artif. Intell., Aug. 2019, pp. 4199–4205.

M. Zhang, X. Wang, M. Zhu, C. Shi, Z. Zhang, and J. Zou, “Robust heterogeneous graph network model for adversarial attacks,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 4363–4370.

S. Grohele, M. Schmidt, H. Sirin, D. Zügner, A. Bojchevski, and S. Günnemann, “Robustness of graph neural networks at scale,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 13992–14001.

H. Yuan, J. Tang, X. Hu, and S. Ji, “XGNN: Towards model-level explanations of graph neural networks,” in Proc. 26th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining, Aug. 2020, pp. 430–438.

L. Zhao, B. Zong, Z. Guan, K. Zhang, and W. Zhao, “Substructure assembling network for graph classification,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 8460–8469.

L. Chu, X. Hu, J. Li, Wang, and J. Pei, “Exact and consistent interpretation for piecewise linear neural networks: A closed form solution,” in Proc. 24th ACM SIGKDD Conf. Knowledge Discovery Data Mining, Jul. 2018, pp. 1244–1253.

Q. Huang, M. Yamada, Y. Tian, D. Singh, D. Yin, and Y. Chang, “GraphLIME: Local interpretable model explanations for graph neural networks,” 2020, arXiv:2001.06216.

M. Bahij, “Robust counterfactual explanations on graph neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 5644–5655.

C. Olah, A. Mordvintsev, and L. Schubert (2017). “Feature Visualization. Distill. [Online]. Available https://distill.pub/2017/visualizing-features.”

J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan, “Learning to explain: An information-theoretic perspective on model interpretation,” in Proc. Int. Conf. Mach. Learn., vol. 80, 2018, pp. 882–890.

D. Duvenaud et al., “Convolutional networks on graphs for learning molecular fingerprints,” in Proc. Adv. Neural Inf. Process. Syst., Dec. 2015, pp. 2224–2232.

Z. Liu, D. Zhou, and J. He, “Towards explainable representation of time-evolving graphs via spatial-temporal graph attention networks,” in Proc. 28th ACM Int. Conf. Knowl. Manage., Nov. 2019, pp. 2137–2140.

E. Dai and S. Wang, “Towards self-explainable graph neural network,” in Proc. 30th ACM Int. Conf. Knowl. Manage., Oct. 2021, pp. 302–311.

Z. Zhang, Q. Liu, H. Wang, C. Lu, and C. Lee, “ProtGNN: Towards self-explanation graph neural networks,” in Proc. AAAI Conf. Artif. Intell., 2022, pp. 1–13.

A. Subramonian, “Motif-driven contrastive learning of graph representations,” in Proc. AAAI Conf. Artif. Intell., 2021, pp. 1–2.

J. Yu, T. Xu, Y. Rong, Y. He, J. Huang, and R. He, “Graph information bottleneck for subgraph recognition,” in Proc. Int. Conf. Learn. Represent., 2021, pp. 1–12.

F. Baldassarre and H. Aitzouz, “Explainability techniques for graph convolutional networks,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 1–21.

R. Henderson, D. Clevett, and F. Montanari, “Improving molecular graph network explainability with orthonormalization and induced sparsity,” in Proc. Int. Conf. Mach. Learn., 2022, pp. 15524–15534.
Laplace mechanism: Differential privacy preservation in deep learning,” in Proc. IEEE Int. Conf. Data Min. (ICDM), Nov. 2017, pp. 385–394.

[250] S. Jamin and D. Gatica-Perez, “Locally private graph neural networks,” in Proc. ACM SIGKDD Conf. Comput. Commun., Nov. 2021, pp. 2310–2315.

[252] D. Xu, S. Yuan, X. Wu, and H. Phan, “DPNE: Differentially private neural embedding,” in Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining, vol. 90, Cham, Switzerland: Springer, 2018, pp. 235–246.

[254] P. Liao et al., “Information obfuscation of graph neural networks,” in Proc. Int. Conf. Mach. Learn., Jul. 2019, pp. 19–30.

[256] K. Li, G. Luo, Y. Ye, W. Li, S. Ji, and Z. Cai, “Adversarial privacy-preserving graph embedding against inference attack,” IEEE Internet Things J., vol. 8, no. 8, pp. 9604–9615, Apr. 2021.

[258] F. Tramèr and D. Boneh, “Slalom: Fast, verifiable and private execution of neural networks in trusted hardware,” in Proc. Int. Conf. Learn. Represent., Mar. 2020, pp. 1–19.

[262] S. Zhang et al., “Federated cross-node training with missing data,” in Proc. Int. Joint Conf. Artif. Intell., Jul. 2020, pp. 4661–4667.

[264] C. Chen et al., “Vertically federated graph neural network for privacy-preserving node classification,” in Proc. Adv. Neural Inf. Process. Syst., Dec. 2020, pp. 12–17.

[266] W. Zhu, J. Luo, and A. White, “Federated learning of molecular properties with graph neural networks in a heterogeneous setting,” 2021, arXiv:2112.01343.

[268] C. G. Ramage and A. Rekik, “A federated multigraph integration approach for connectional brain template learning,” in Proc. Int. Workshop Multimodal Learn. Clin. Decis. Support, vol. 13050. Cham, Switzerland: Springer, 2021, pp. 36–47.

[270] L. Shen, X. Chen, J. Shi, Y. Dong, and B. Fang, “An efficient 3-party framework for privacy-preserving neural network inference,” in Proc. EURASIP J. Softw. Eng. Comput. Sci. Appl., vol. 12308. Cham, Switzerland: Springer, 2020, pp. 16337–16346.

[272] L. Zhu, Z. Liu, and S. Han, “Deep leakage from model response,” in Proc. IEEE Conf. Comput. Secur. (ICCSP), Apr. 2020, pp. 50–60, May 2020.

[274] J. Kang and H. Tong, “Fair graph mining,” in Proc. 33rd Conf. Neural Inf. Process. Syst., Dec. 2020, pp. 2739–2745.

[276] H. Ling, Z. Jiang, L. Yao, S. Ji, and N. Zou, “Learning fair graph representations via automated data augmentation,” in Proc. Int. Conf. Learn. Represent., 2023, pp. 1–12.

[278] K. Song, H. Park, J. Lee, A. Kim, and J. Jung, “COVID-19 infection inference with graph neural networks,” Sci. Rep., vol. 13, no. 1, p. 11,649, Jul. 2023.

[280] W. Fan et al., “Graph neural networks for social recommendation,” in Proc. World Wide Web Conf., New York, NY, USA, May 2019, pp. 417–426.

[282] R. Zemel, “Fairness through awareness,” in Proc. R. Soc. Proc. A, vol. 472. The Royal Society, Feb. 2019, pp. 20150260–20150260.

[284] J. Kang and H. Tong, “Fair graph mining,” in Proc. 30th ACM Int. Conf. Knowl. Manage., Oct. 2021, pp. 4849–4852.

[286] J. K. Song, H. Kang, and H. Tong, “Individual fairness for graph neural networks: A ranking based approach,” in Proc. ACM Conf. Knowl. Discovery Data Mining (SIGKDD), 2022, pp. 300–310.

[288] F. Kamiran and R. Zafar, “Explainable and non-explainable discrimination in classification,” in Discrimination and Privacy in the Information Society (Studies in Applied Philosophy, Epistemology and Rational Ethics), vol. 3. Cham, Switzerland: Springer, 2013, pp. 155–170.

[290] J. Chen, N. Kallus, X. Mao, G. Svacha, and J. Kang, “Privacy-preserving collaborative machine learning,” in Proc. Eur. Symp.Resp. Comput. Syst., vol. 12, 2020, pp. 6600–6610.

[292] G. Luo, B. Feng, G. C. Fox, and L. Jiang, “GlypF: Fast and accurately training deep neural networks on encrypted data,” in Proc. Adv. Neural Inf. Process. Syst., 2022, pp. 99–106.

[294] M. J. Kusner, J. Loftus, C. Russell, and R. Silva, “Counterfactual fairness,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 4066–4076.

[296] S. Verma and J. Rubin, “Fairness definitions explained,” in Proc. IEEE/ACM Int. Workshop Softw. Fairness (FairWare), May 2018, pp. 1–7.

[298] C. Dwork, M. Hardt, P. Pitassi, E. Reingold, and R. Zemel, “Fairness through awareness,” in Proc. 3rd Innov. Theor. Comput. Sci. Conf. New York, NY, USA: Association for Computing Machinery, Jan. 2012, pp. 214–226.

[300] H. Xie, J. Li, X. Wang, and C. Yang, “Federated graph classification over non-IID graphs,” in Proc. Adv. Neural Inf. Process. Syst., 2021, pp. 18839–18852.

[302] J. Kang and H. Tong, “Fair graph mining,” in Proc. 30th ACM Int. Conf. Knowl. Manage., Oct. 2021, pp. 4849–4852.

[304] S. A. Saxena, K. Hui, and S. Lee, “How do fairness definitions fare: Examining public attitudes towards algorithmic definitions of fairness,” in Proc. AAAI Conf. AI, Ethics, Soc., Jan. 2019, pp. 99–106.

[306] T. Rahman, B. Surma, M. Backes, and Y. Zhang, “FairWalk: Towards fair graph embedding,” in Proc. 28th Int. Joint Conf. Artif. Intell., Aug. 2019, pp. 3289–3295.

[308] M. J. Kusner, J. Loftus, C. Russell, and R. Silva, “Counterfactual fairness,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 4066–4076.
with heterophily: A survey," 2022, arXiv:2201.02967.

[302] S. Corbett-Davies, E. Pierson, A. Feller, S. Goel, and A. Huq, "Algorithmic decision making and the cost of fairness," in Proc. 23rd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, Aug. 2017, pp. 797–806.

[303] J. Kang, Y. Zhu, Y. Xia, J. Luo, and H. Tong, "RawGNN: Towards rawkival difference principle on graph convolutional network," in Proc. ACM Web Conf., May 2020, pp. 1214–1225.

[304] Y. Dong, S. Wang, J. Ma, N. Liu, and J. Li, "Interpreting unfairness in graph neural networks via training node attribution," in Proc. AAAI Conf. Intell. Inform. Syst., 2020, pp. 552–560.

[305] T. Xie, Y. Ma, J. Kang, H. Tong, and R. Maciejewski, "FairRankVs1: A visual analytics framework for exploring algorithmic fairness in graph mining models," IEEE Trans. Vis. Comput. Graphics, 2028, no. 1, pp. 368–377, Jan. 2022.

[306] H. Husein et al., "Adversarial inter-group link injection degrades the fairness of graph neural networks," in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2022, pp. 975–980.

[307] R. Wang, X. Wang, C. Shi, and L. Song, "Uncovering the structural fairness in graph contrastive learning," Represent. Evol. Comput., vol. 1, no. 1, pp. 1–21, Mar. 2023.

[308] W. Song, Y. Dong, N. Liu, and J. Li, "GUIDE: Group equality informed individual fairness in graph neural networks," in Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining, Aug. 2022, pp. 1625–1634.

[309] Y. Wang, W. Ma, M. Zhang, Y. Liu, and S. Ma, "A survey on the fairness of recommender systems," ACM Trans. Inf. Syst., vol. 41, no. 3, pp. 1–43, Jul. 2023.

[310] D. Jin et al., "A survey on fairness-aware recommender systems," Inf. Fusion, vol. 100, Dec. 2023, Art. no. 107216. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566225320002221

[311] M. Naghiaei, H. A. Rahmani, and Y. Deldigoj, “CPFair: Personalized consumer and producer fairness-aware recommender systems,” in Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2022, pp. 770–779.

[312] C.-T. Li, C. Hsu, and Y. Zhang, "FairSR: Fairness-aware social recommendation through multi-task learning with preference graph embeddings," ACM Trans. Intell. Syst. Technol., vol. 13, no. 1, pp. 1–21, Feb. 2022.

[313] Y. Zhang et al., “Causal intervention for leveraging popularity bias in recommendation,” in Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2021, pp. 11–20.

[314] Y. Ge et al., “Towards long-term recommendation in recommendation,” in Proc. 14th ACM Int. Conf. Web Search Data Mining, Mar. 2021, pp. 445–453.

[315] D. Cheng, Y. Tu, Z. Ma, N. Li, and Z. Zhang, “Risk assessment for networked guarantee loans using high-order graph attention representation,” in Proc. 28th Int. Joint Conf. Artif. Intell., Aug. 2019, pp. 5882–5828.

[316] T. Liang et al., “Credit risk and limited forecasting in e-commerce consumer lending service via multi-view-aware mixture-of-experts nets,” in Proc. 14th ACM Int. Conf. Web Search Data Mining, Jerusalem, Israel, Mar. 2021, pp. 229–237.

[317] S. X. Rao et al., “Flexible explains: Explainable fraud transaction detection,” Proc. VLDB Endowment, vol. 15, no. 3, pp. 427–436, Nov. 2021.

[318] Y. Liu et al., “Anomaly detection in dynamic graphs via transductive self-supervised learning,” Knowl. Data Eng., vol. 35, no. 12, pp. 12081–12094, Dec. 2023.

[319] Z. Zheng et al., “Graph neural networks for graphs
ABOUT THE AUTHORS

He Zhang is currently working the Ph.D. degree at the Faculty of Information Technology, Monash University, Clayton, VIC, Australia.

He has a profound interest in graph learning and the development of trustworthy AI systems. His research in trustworthy graph neural networks has led to several academic publications in top journals and conferences, including IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), IJCL, NDSS, AAAI, and CIKM.

Bang Wu received the Ph.D. degree from the Faculty of Information Technology, Monash University, Clayton, VIC, Australia, in 2023.

His research interests include trustworthy graph-based machine learning, trustworthy machine learning on multimodal systems, and various facets of security and privacy in machine learning across different domains.
Xingliang Yuan (Senior Member, IEEE) is currently an Associate Professor with the School of Computing and Information Systems, The University of Melbourne, Parkville, VIC, Australia. His research has been supported by the Australian Research Council, CSIRO, Australian DHA, MRFF, and the Oceania Cyber Security Centre. His work has appeared in prestigious venues in computer security, computer networks, and distributed systems, such as IEEE S&SP, ACM CCS, NDSS, IEEE INFOCOM, ICML, IEEE TDSC, IEEE TIFS, and IEEE TPDS. His research interests include data security and privacy, machine learning security and privacy, and secure networked systems.

Dr. Yuan was the sole recipient of the Dean’s Award for Excellence in Research by an Early Career Researcher at the Monash Faculty of Information Technology (IT) in 2020. He was a co-recipient of the Best Paper Award at the European Symposium on Research in Computer Security (ESORICS) 2021. He has been the Track Co-Chair of IEEE ICDCS 2024 (security, privacy, and trust) and the Program Co-Chair of NSS 2022 and SecTI 2023. He is an Editor of IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING.

Hanghang Tong (Fellow, IEEE) received the M.Sc. and Ph.D. degrees in machine learning from Carnegie Mellon University, Pittsburgh, PA, USA, in 2008 and 2009, respectively.

He was an Associate Professor with the School of Computing, Informatics, and Decision Systems Engineering (CIDSE), Arizona State University, Tempe, AZ, USA. He is currently an Associate Professor with the Department of Computer Science, University of Illinois at Urbana–Champaign, Urbana, IL, USA. He has published over 100 refereed articles. His research interests include large-scale data mining for graphs and multimedia.

Dr. Tong has served as a Program Committee Member in data mining, database, and artificial intelligence venues (e.g., SIGKDD, SIGMOD, AAAI, WWW, and CIKM). He received several awards, including the SDM/IBM Early Career Data Mining Research Award in 2018, the NSF CAREER Award in 2017, the ICDM 10-Year Highest Impact Paper Award in 2015, and four Best Paper Awards at TUP 2014, CIKM 2012, SDM 2008, and ICDM 2006. He is the Editor-in-Chief of SIGKDD Explorations (ACM), an Action Editor of Data Mining and Knowledge Discovery (Springer), and an Associate Editor of Knowledge and Information Systems (Springer).

Jian Pei (Fellow, IEEE) is currently a Professor with Duke University, Durham, NC, USA. His research interests lie in the general areas of data science, big data, data mining, and database systems. Since 2000, he has published one textbook, two monographs, and over 300 research papers in refereed journals and conferences, which have been cited extensively by others. His research has generated a remarkable impact substantially beyond academia. For example, his algorithms have been adopted by industry in production and popular open-source software suites. He is one of the most cited authors in data mining, database systems, and information retrieval. He also demonstrated outstanding professional leadership in many academic organizations and activities. He maintains a wide spectrum of industry relations with both global and local industry partners. He is an active consultant and coach for the industry.

Dr. Pei was recognized as a Fellow of the Royal Society of Canada (Canada’s National Academy), the Canadian Academy of Engineering, and ACM. He received many prestigious awards, including the 2017 ACM SIGKDD Innovation Award, the 2015 ACM SIGKDD Service Award, the 2014 IEEE ICDM Research Contributions Award, the British Columbia Innovation Council 2005 Young Innovator Award, the NSERC 2008 Discovery Accelerator Supplements Award, the IBM Faculty Award in 2006, the KDD Best Application Paper Award in 2008, the ICDE Influential Paper Award in 2018, the PAKDD Best Paper Award in 2014, and the PAKDD Most Influential Paper Award in 2009. He was the Chair of ACM SIGKDD from 2017 to 2021 and the general co-chair or the program committee co-chair of many premier conferences. He was the Editor-in-Chief of IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE) from 2013 to 2016.

Shirui Pan (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Technology Sydney (UTS), Ultimo, NSW, Australia, in 2015.

He was a Senior Lecturer with the Faculty of Information Technology (IT), Monash University, Clayton, VIC, Australia. He is currently a Professor with the School of Computing and Information Technology, Griffith University, Southport, QLD, Australia. To date, he has published over 200 research papers in top-tier journals and conferences, including IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS), ICML, NeurIPS, KDD, AAAI, IJCAI, WWW, CVPR, and ICDM. His research interests include data mining and machine learning, with a focus on graph neural networks (GNNs).

Dr. Pan is an ARC Future Fellow and a Fellow of the Queensland Academy of Arts and Sciences (FQA). His GNN research received the 2024 CIS IEEE TNNLS Outstanding Paper Award and the 2020 IEEE ICDM Best Student Paper Award. He is recognized as one of the AI 2000 AAAI/IJCAI Most Influential Scholars in Australia.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.