A Survey of Graph Neural Networks in Real World: Imbalance, Noise, Privacy and OOD Challenges

Wei Ju, *Member, IEEE*, Siyu Yi, *Member, IEEE*, Yifan Wang, Zhiping Xiao, Zhengyang Mao, Hourun Li, Yiyang Gu, Yifang Qin, Nan Yin, Senzhang Wang, *Member, IEEE*, Xinwang Liu, *Senior Member, IEEE*, Philip S. Yu, *Life Fellow, IEEE*, and Ming Zhang, *Member, IEEE*

(Survey Paper)

Abstract—**Graph-structured data exhibits universality and widespread applicability across diverse domains, such as social network analysis, biochemistry, financial fraud detection, and network security. Significant strides have been made in leveraging Graph Neural Networks (GNNs) to achieve remarkable success in these areas. However, in real-world scenarios, the training environment for models is often far from ideal, leading to substantial performance degradation of GNN models due to various unfavorable factors, including imbalance in data distribution, the presence of noise in erroneous data, privacy protection of sensitive information, and generalization capability for out-of-distribution (OOD) scenarios. To tackle these issues, substantial efforts have been devoted to improving the performance of GNN models in practical real-world scenarios, as well as enhancing their reliability and robustness. In this paper, we present a comprehensive survey that systematically reviews existing GNN models, focusing on solutions to the four mentioned real-world challenges including imbalance, noise, privacy, and OOD in practical scenarios that many existing reviews have not considered. Specifically, we first highlight the four key challenges faced by existing GNNs, paving the way for our exploration of real-world GNN models. Subsequently, we provide detailed discussions on these four aspects, dissecting how these solutions contribute to enhancing the reliability and robustness of GNN models. Last but not least, we outline promising directions and offer future perspectives in the field.**

Index Terms—**Graph Neural Networks, Imbalance, Noise, Privacy, Out-of-Distribution.**

Received 7 March 2024; revised 25 July 2025; accepted 4 November 2025. Date of publication 7 November 2025; date of current version 4 February 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62306014 and Grant 62276002, in part by the National Science Foundation under Grant III-2106758 and Grant POSE-2346158, in part by the Postdoctoral Fellowship Program (Grade A) of CPSF under Grant BX20250376 and Grant BX20240239, in part by China Postdoctoral Science Foundation under Grant 2024M762201, in part by Sichuan Science and Technology Program under Grant 2025ZNSFSC1506 and Grant 2025ZNSFSC0808, and in part by Sichuan University Interdisciplinary Innovation Fund . Recommended for acceptance by A. Sharma. *(Corresponding authors: Siyu Yi; Ming Zhang.)*

Wei Ju is with the College of Computer Science, Sichuan University, Chengdu 610207, China (e-mail: juwei@scu.edu.cn).

Siyu Yi is with the College of Mathematics, Sichuan University, Chengdu 610207, China (e-mail: siyuyi@scu.edu.cn).

Yifan Wang is with the School of Information Technology & Management, University of International Business and Economics, Beijing 100029, China (e-mail: yifanwang@uibe.edu.cn).

Zhiping Xiao is with the Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, WA 98195 USA (e-mail: patxiao@uw.edu).

Zhengyang Mao, Hourun Li, Yiyang Gu, Yifang Qin, and Ming Zhang are with the School of ComputerScience, Peking University, Beijing 100871, China (e-mail: zhengyang.mao@stu.pku.edu.cn; lihourun@stu.pku.edu.cn; yiyanggu@pku.edu.cn; qinyifang@pku.edu.cn; mzhang_cs@pku.edu.cn).

Nan Yin is with the Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi, United Arab Emirates (e-mail: yinnan8911@gmail.com).

Senzhang Wang is with the School of Computer Science and Technology, Central South University, Changsha 410083, China (e-mail: szwang@csu.edu.cn).

Xinwang Liu is with the College of Computer, National University of Defense Technology, Changsha 410073, China (e-mail: xinwangliu@nudt.edu.cn).

Philip S. Yu is with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: psyu@uic.edu).

Digital Object Identifier 10.1109/TPAMI.2025.3630673

I. Introduction

GRAPH-STRUCTURED data, characterized by nodes and edges that represent interconnected entities and relationships, possesses inherent complexity and versatility. The interconnected nature of graphs allows them to model a wide range of real-world scenarios where entities and their interactions play a crucial role. Analyzing graph data is of paramount importance as it enables us to gain insights into intricate patterns, uncover hidden structures, and understand the dynamics of interconnected systems [1], [2]. The applicability of graph data extends across various domains; for instance, in social network analysis, graphs can represent relationships between individuals [3], in bioinformatics, molecular structures can be modeled as graphs [4], and transportation networks can be also expressed as graphs to optimize routes and logistics [5].

Recently, the landscape of graph data analysis has been significantly shaped by the widespread adoption and remarkable success of Graph Neural Networks (GNNs) [6], [7], [8], [9]. GNNs have emerged as a cornerstone in graph learning, demonstrating exceptional performance in various applications. The fundamental idea behind GNNs lies in their ability to capture complex relationships within graph-structured data by iteratively aggregating and updating information from neighboring nodes [10]. This enables GNNs to learn meaningful representations of nodes, capturing both local and global patterns within the graph [1]. The versatility and effectiveness of GNNs are prominently demonstrated in various real-world applications. In e-commerce, platforms like Alibaba [11] leverage GNNs to comprehend user behavior, thereby enabling personalized product recommendations and enhancing overall user engagement. Social media such as Pinterest [12] utilize GNNs for content recommendation, successfully connecting users with relevant

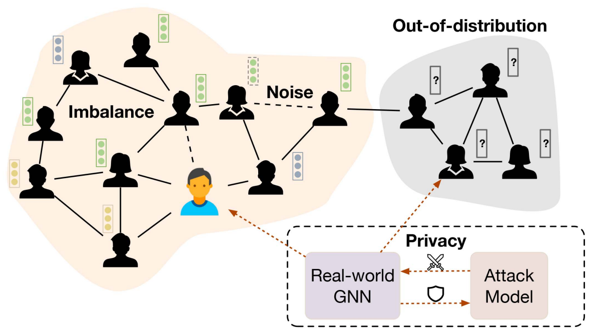

Fig. 1. An illustrative example of GNN models handling practical social network scenarios. User data extracted from real-world platforms typically exhibit long-tailed distributions, indicating there are widespread mainstream user types alongside lots of rare genres. The interactions among users may be influenced by structural noises and the presence of fake labels. Moreover, the practical GNN models are confronted with attack models and user information leakage issues. The generalization of models from existing business scenarios to novel environments also introduces OOD concerns. .

and appealing content. Additionally, GNNs achieve remarkable success in scenarios such as simulating complex physical systems [13], [14] and accelerating drug discovery processes [15], [16].

Despite the outstanding performance exhibited by current GNN models, it is crucial to acknowledge that their training typically occurs within an idealized environment, where the training data is clean, standardized, and comprehensive. However, in real-world scenarios, GNN models typically face various challenges that significantly compromise their performance and may even lead to model collapse [17], [18]. This discrepancy between idealized training conditions and real-world challenges poses a critical issue in the deployment of GNNs. For example, in the fraud detection of financial transactions [19], compared with non-fraud cases, the scarcity of fraud cases leads to an imbalance dataset. Due to this **imbalance** problem, GNNs may have difficulty effectively learning patterns related to fraud. In bioinformatics [20], experimental errors or anomalies may introduce **noise**, making it difficult for GNNs to accurately predict molecular structures or identify potential patterns in biological data. In social network analysis [21], GNNs models needs to strike a balance between extracting valuable information from the network and preserving user **privacy**. Furthermore, in network security [22], GNNs used to detect network threats may encounter struggle when facing novel and previously unseen **out-of-distribution (OOD)** attacks. The illustrative example in Fig. 1 further elucidates the challenges encountered in real-world social network scenarios.

To cope with the myriad challenges that GNN models face in real-world scenarios, researchers have dedicated significant efforts to address these adverse factors. To comprehensively and systematically summarize the methodologies employed in real-world scenarios, we present a thorough survey in this paper. This survey primarily focuses on the solutions devised for GNN models confronting four prevalent real-world conditions: *Imbalance, Noise, Privacy, and Out-of-Distribution*. By consolidating existing research endeavors, this survey aims to provide a comprehensive overview of the current landscape. Additionally, we aim to present prospective research frontiers that can guide researchers in reviewing, summarizing, and formulating future strategies to enhance the reliability and robustness of GNN models in practical applications.

Differences between this survey and existing ones. Until now, there have been several other literature reviews that have delved into real-world GNNs from different aspects [17], [18], [23], [24], [25], and they are closely relevant to our research. While these surveys share relevance with our work, they also exhibit distinctions in their specific focuses. For example, Wu et al. [23] focus on three aspects of GNN models: reliability, explainability, and privacy. Dai et al. [18] conduct a more detailed discussion covering privacy, robustness, fairness, and explainability. Zhang et al. [17], building upon the foundation laid by [18], explore the emerging topics of accountability and environmental well-being. These three concurrent works center their themes around trustworthy GNNs, approaching from the perspective of creating more reliable AI systems. Different from theses works, our survey originates from real-world considerations, concentrating on practical scenarios. Further, Oneto et al. [24], expanding on the trustworthy foundation, encompass more macroscopic elements such as automated operations with guarantees on graphs, aiming for more intelligent and responsible GNN models. To the best of our knowledge, the survey most closely related to ours is [25], which summarizes reliable graph learning from the aspects of inherent noise, distribution shift, and adversarial attack. Besides these, our survey also address the prevalent issues of data imbalance and privacy in real-world scenarios. It is worth noting that their survey [25] only covers methods up to 2022, lacking coverage of the latest developments in the past three years.

Our Contribution: This survey aims to comprehensively summarize the advancements of GNN models in the real world while also paving the way for future exploration. It serves as a valuable resource for both researchers and practitioners by providing them with an overview and the latest developments in GNNs in practical scenarios. The key contributions of this survey are highlighted below:

- *Systematic Taxonomy:* A novel taxonomy is proposed to systematically categorize existing real-world GNN models, focusing primarily on the models to address imbalance, noise, privacy, and out-of-distribution issues, and presenting representative methods.
- *Extensive Review:* For each category covered in this survey, we summarize its basic principles and components, and provide detailed insights into representative algorithms, followed by a systematic discussion of their findings.
- *Future Perspectives:* We identify limitations and confront challenges associated with current real-world GNN models, and outline potential research directions, offering a novel perspective on future avenues of study.

II. Taxonomy

To gain deeper insights of GNN models in real-world scenarios, we have spotlighted key research efforts, delved into their

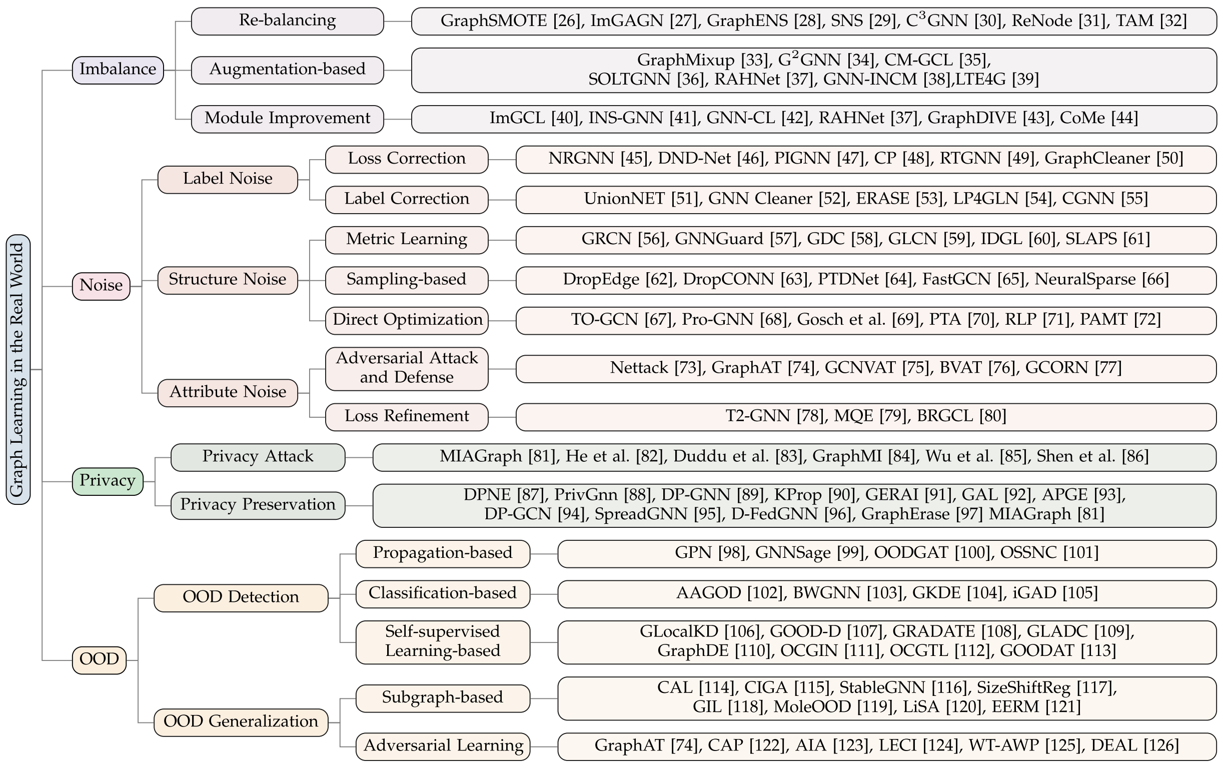
Graph Learning in the Real World
Imbalance
Re-balancing
GraphSMOTE [26], ImGAGN [27], GraphENS [28], SNS [29], C^{3}GNN [30], ReNode [31], TAM [32]
Augmentation-based
GraphMixup [33], G^{2}GNN [34], CM-GCL [35], SOLTGNN [36], RAHNet [37], GNN-INCM [38],LTE4G [39]
Module Improvement
ImGCL [40], INS-GNN [41], GNN-CL [42], RAHNet [37], GraphDIVE [43], CoMe [44]
Noise
Label Noise
Loss Correction
NRGNN [45], DND-Net [46], PIGNN [47], CP [48], RTGNN [49], GraphCleaner [50]
Label Correction
UnionNET [51], GNN Cleaner [52], ERASE [53], LP4GLN [54], CGNN [55]
Structure Noise
Metric Learning
GRCN [56], GNNGuard [57], GDC [58], GLCN [59], IDGL [60], SLAPS [61]
Sampling-based
DropEdge [62], DropCONN [63], PTDNet [64], FastGCN [65], NeuralSparse [66]
Direct Optimization
TO-GCN [67], Pro-GNN [68], Gosch et al. [69], PTA [70], RLP [71], PAMT [72]
Attribute Noise
Adversarial Attack and Defense
Nettack [73], GraphAT [74], GCNVAT [75], BVAT [76], GCORN [77]
Loss Refinement
T2-GNN [78], MQE [79], BRGCL [80]
Privacy
Privacy Attack
MIAGraph [81], He et al. [82], Duddu et al. [83], GraphMI [84], Wu et al. [85], Shen et al. [86]
Privacy Preservation
DPNE [87], PrivGnn [88], DP-GNN [89], KProp [90], GERAI [91], GAL [92], APGE [93], DP-GCN [94], SpreadGNN [95], D-FedGNN [96], GraphErase [97] MIAGraph [81]
OOD
OOD Detection
Propagation-based
GPN [98], GNNSage [99], OODGAT [100], OSSNC [101]
Classification-based
AAGOD [102], BWGNN [103], GKDE [104], iGAD [105]
Self-supervised Learning-based
GLocalKD [106], GOOD-D [107], GRADATE [108], GLADC [109], GraphDE [110], OCGIN [111], OCGTL [112], GOODAT [113]
OOD Generalization
Subgraph-based
CAL [114], CIGA [115], StableGNN [116], SizeShiftReg [117], GIL [118], MoleOOD [119], LiSA [120], EERM [121]
Adversarial Learning
GraphAT [74], CAP [122], AIA [123], LECI [124], WT-AWP [125], DEAL [126]

Fig. 2. An overview of the taxonomy for existing GNN models in real world.

motivations, and concisely summarized their primary technical contributions. The overall structure of the paper is depicted in Fig. 2. This survey establishes a novel taxonomy, categorizing these works into four distinct classes: *Imbalance, Noise, Privacy, and Out-Of-Distribution*. These categories serve as a comprehensive framework for reviewing and analyzing these works across diverse scenarios. We will provide a brief overview of these four real-world factors:

- *Imbalance* in graph data refers to the uneven distribution of class labels in a graph [26]. Solving this problem is crucial for preventing the learning process from being biased towards the majority class, which will hinder the model's ability to capture minority class patterns. To alleviate this situation, three main strategies are usually employed. The first is the re-balancing strategy, which aims to reduce class dominance [26], [31] by adjusting sample distributions or loss functions. The second strategy is augmentation-based, which aims to support model training by introducing extra data or structural information [34], [39]. The last strategy is module enhancement [40], [43] to increase the model's ability to learn discriminative features under imbalance.
- *Noise* in graph data denotes incorrect, irrelevant, or misleading information that can impair GNN performance [45], generally classified into three types: *Label Noise*, *Structure Noise*, and *Attribute Noise*. The first is caused by label allocation errors due to annotation errors or inconsistent data, which is typically addressed through loss correction and label correction [45], [52]. The second arises from inaccurate or missing edges and distorted graph topology, which is commonly mitigated through metric learning, sampling-based methods and direct optimization [59], [62], [67]. The third arises from manual entry errors or intentional manipulation, typically addressed by ensuring robust representation learning through adversarial training [73] and loss refinement [79].
- *Privacy* in graph data involves protecting sensitive information tied to nodes or edges to ensure confidentiality during training and inference processes [127]. Since graph learning usually deals with personal or confidential data, protecting privacy is of crucial importance. The solution aims to balance practicality and protection, and is usually divided into *Privacy Attack* and *Privacy Preservation*. Privacy attack uses the model or data privacy loophole to reveal hidden information [81], [83], [85]. And privacy preservation is focused on by adversarial learning, the federated training, and other technical defense [93], [94], [128].
- *Out-of-distribution (OOD)* in graph data are instances that differ significantly from the distribution seen during training [129]. In graph learning, OOD scenarios require the model to recognize and adapt to unseen graph instances. Solving this problem involves two key tasks: *OOD Detection* and *OOD Generalization*. OOD detection [98], [107], [110] focuses on the recognition beyond the distribution of the training data points, usually use anomaly detection or uncertainty estimation techniques. Instead, OOD generalization [114], [130] aims to improve the

robustness of the models, to new, unseen graph for accurate prediction.

III. Preliminary

A. Graph

Given a graph denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \dots, v_{|\mathcal{V}|}\}$ is the node set and $\mathcal{E}$ is the edge set which can be represented by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where $\mathbf{A}_{ij} = 1$ if $(v_i, v_j) \in \mathcal{E}$, otherwise $\mathbf{A}_{ij} = 0$. Each node v_i in the graph is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^d$, constituting the feature matrix of graph $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d}$, where d presents the number of dimensions in the features. Therefore, we can also represent a graph as $\mathcal{G} = \{\mathbf{X}, \mathbf{A}\}$, in which $\mathbf{Y}$ denotes the label vector for the labeled nodes or graphs.

B. Graph Neural Networks

Graph Neural Networks (GNNs) [6], [7], [8] represent a class of neural network architectures specifically tailored for learning representations of the graph's components—its nodes, edges and even entire graph—to capture the complex relationships and structures within the graph. A central mechanism of GNNs is the message-passing paradigm [10], where the embedding of a node v_i is iteratively updated through interactions with its neighbors, denoted as:

$$\begin{aligned}\mathbf{h}_i^{(l)} &= \mathrm{GNN}^{(l)}\left(\mathbf{h}_i^{(l-1)}, \left\{\mathbf{h}_j^{(l-1)}\right\}_{v_j\in\mathcal{N}(v_i)}\right) \\ &= \mathcal{C}^{(l)}\left(\mathbf{h}_i^{(l-1)}, \mathcal{A}^{(l)}\left(\left\{\mathbf{h}_j^{(l-1)}\right\}_{v_j\in\mathcal{N}(v_i)}\right)\right), \qquad (1)\end{aligned}$$

where $\mathbf{h}_i^{(l)}$ indicates the embedding of v_i at layer $l \in \{1, \dots, L\}$ and $\mathcal{N}(v_i)$ denotes the neighbors of v_i derived from $\mathbf{A}$. $\mathcal{A}^{(l)}$ and $\mathcal{C}^{(l)}$ are the message aggregating and embedding updating functions at layer l, respectively. Finally, the node-level representation is $\mathbf{h}_i = \mathbf{h}_i^{(L)}$ at layer L, while the graph-level representation can be attained by a READOUT aggregation function, defined as:

$$\mathbf{h}_{\mathcal{G}} = \mathrm{READOUT}\left(\left\{\mathbf{h}_i^{(L)}\right\}_{v_i\in\mathcal{V}}\right), \qquad (2)$$

where READOUT could be averaging or other graph-level pooling functions depending on the model [131], [132], [133].

C. Computational Tasks

Computational tasks related to graphs can generally be classified into two primary categories: node-level and graph-level tasks. At the node level, the main tasks involve classifying nodes [134], ranking nodes [135], clustering nodes [136] and predicting links between nodes [137]. On the other hand, tasks at the graph level primarily include classifying entire graphs [138], [139], matching different graphs [140] and generating new graphs [141]. Here we introduce three representative computational tasks on the graph.

Node Classification: Given graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with a set of labeled nodes denoted as $\mathcal{V}_{\mathrm{L}} \subset \mathcal{V}$ and unlabeled nodes set denoted as $\mathcal{V}_{\mathrm{U}} = \mathcal{V} \setminus \mathcal{V}_{\mathrm{L}}$. We assume each node $i \in \mathcal{V}_{\mathrm{L}}$ is associated with a label y_i. Node classification aims to learn a model using $\mathcal{G}$ and its available label set to predict the labels of the unlabeled nodes in $\mathcal{V}_{\mathrm{U}}$.

Link Prediction: Given graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, with a set of known edges $\mathcal{E}_{\mathrm{K}}$. We assume the existence of unobserved links $\mathcal{E}_{\mathrm{U}} = \mathcal{E} \setminus \mathcal{E}_{\mathrm{K}}$ between nodes. Link prediction aims to learn a model from $\mathcal{G}$ to predict whether a link exists between nodes v_i and v_j, where $(v_i, v_j) \notin \mathcal{E}_{\mathrm{K}}$.

Graph Classification: Beyond tasks focused on individual nodes, graph classification operates at the level of entire graphs. Given a training graph dataset $\mathcal{D} = \{(\mathcal{G}_i, y_i)\}_{i=1}^{|\mathcal{D}|}$ with multiple graphs, we assume each graph instance belongs to a certain class, where y_i is the label of graph $\mathcal{G}_i$ and $|\mathcal{D}|$ represents the total number of graphs in the training set. Graph classification aims to learn a model from the dataset $\mathcal{D}$ to predict the labels of unseen test graphs.

Based on the basic concept of graphs, more details about GNN applications in real-world practical scenarios can be found in the following sections.

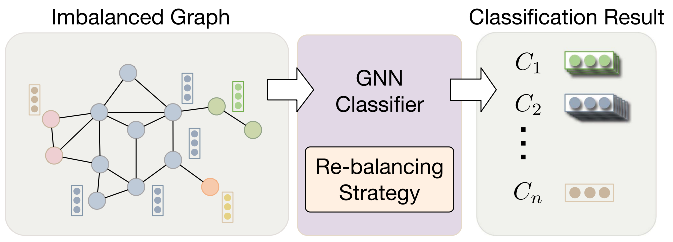

Fig. 3. Illustration of the data imbalanced problem. The labels assigned to nodes or graphs that obtained from real-world data sources always suffer from severe class imbalance issue brought by the long-tail distribution of samples. The challenge calls for various applicable re-balancing strategies to train robust and reliable GNNs.

IV. Imbalance

In real-world GNN applications, data imbalance poses a major challenge, often showing significant disparities in class instance counts. This is common in tasks like fraud detection [142] and anomaly detection [143]. The dominance of majority classes leads to minority classes being underrepresented, harming overall performance. Formally, let $\{C_1, C_2, \dots, C_K\}$ denote K classes with $|C_k|$ samples in class k, satisfying $|C_1| \geq |C_2| \geq \cdots \geq |C_K|$. The goal is to train a GNN classifier that performs well on both majority (e.g., C_1) and minority (e.g., C_K) classes. In other words, we expect that the trained GNN classifier $\mathcal{F}^*$ can maximize the utility, such as the accuracy (ACC) for each class, i.e.,

$$\mathcal{F}^* = \arg\max_{\mathcal{F}} \left\{\mathrm{ACC}\left(\mathcal{F}(C_k)\right)\right\}_{k=1}^{K}. \qquad (3)$$

To tackle this, existing methods are broadly grouped into three categories: *re-balancing*, *augmentation-based*, and *module improvement* methods. Fig. 3 illustrates the concept, and Table I summarizes representative works. Next, we delve into each strategy, offering a comprehensive overview.

TABLE I
OVERVIEW OF METHODS FOR LEARNING FROM IMBALANCED GRAPHS, CATEGORIZED INTO THREE MAIN TYPES: RE-BALANCING, AUGMENTATION-BASED, AND MODULE IMPROVEMENT METHODS.

Method	Task Type	Re-balancing		Augmentation		Module Improvement		
		RS	CSL	TL	IA	RL	CT	ME
GraphSMOTE [26]	Node-level	✓						
ImGAGN [27]	Node-level	✓						
GraphENS [28]	Node-level	✓						
SNS [29]	Node-level	✓				✓		
C^3GNN [30]	Graph-level	✓				✓		
DataDec [144]	Graph-level	✓						
ReNode [31]	Node-level		✓					
TAM [32]	Node-level		✓					
SOLTGNN [36]	Graph-level			✓				
RAHNet [37]	Graph-level			✓		✓	✓	
GNN-INCM [38]	Node-level			✓				✓
LTE4G [39]	Node-level			✓				✓
GraphMixup [33]	Node-level				✓			
G^2GNN [34]	Graph-level				✓	✓		
CM-GCL [35]	Node-level		✓		✓			
INS-GNN [41]	Node-level					✓		
ImGCL [40]	Node-level	✓				✓		
GNN-CL [42]	Node-level					✓		
GraphDIVE [43]	Graph-level							✓
CoMe [44]	Graph-level		✓			✓		✓
PASTEL [146]	Node-level				✓	✓		
QTIAH-GNN [143]	Node-level		✓			✓		

A. Re-Balancing Approaches

Re-balancing approaches aim at addressing the problem of uneven distribution of training samples across different classes, including two main categories of methods: *re-sampling*, and *cost-sensitive learning*.

Re-sampling (RS): RS adjusts the selection of samples during training. Standard RS techniques involve either replicating samples in the minority class or reducing samples in the majority class. However, in cases of severe imbalance, they can either cause overfitting or weaken performance, respectively. Therefore, recent studies mainly aim to synthesize ($\mathcal{B}_{\text{SYN}}$) minority samples, or partition ($\mathcal{B}_{\text{PAR}}$) the majority samples to balance ($\mathcal{B}$) the classes, formulated as:

$$\mathcal{B}(C_1, \ldots, C_n) = \{\mathcal{B}_{\text{SYN}}(C_{\text{minority}}), \mathcal{B}_{\text{PAR}}\left(C_{\text{majority}}\right)\}, \quad (4)$$

where C_{minority} and C_{majority} contain samples from the majority and minority classes, respectively. GraphSMOTE [26] employs synthetic minority oversampling within the embedding space to increase the representation of minority classes. Moreover, it integrates an edge generator to create new connections between synthesized samples and existing ones, which can produce dependable relational data among samples. To improve sample quality, ImGAGN [27] introduces a generative adversarial graph network that generates synthetic minority nodes and uses a GCN-based discriminator to distinguish real nodes. Despite effectiveness, they still struggle with neighbor memorization under severe imbalance. GraphENS [28] addresses this by generating minority nodes along with their one-hop neighbors, synthesizing complete ego networks. SNS [29] tackles imbalanced heterogeneous networks by adaptively selecting neighbors and enriching the network with synthetic nodes. DataDec [144] proposes a dynamic sparsity framework—data decantation—that selects informative samples via gradient score ranking. Recently, C^3 GNN [30] proposes a cluster-guided contrastive framework, which alleviates imbalance by dividing majority classes into balanced subclasses and enriching them via mixup.

Cost-sensitive Learning (CSL): CSL adjusts training loss for various classes to tackle the imbalances in training. A widely used method involves applying the frequencies of labels from the training data to adjust the weights of the loss function. As a variation of this technique, class-balanced loss (ℓ_{CB}) [145] involves scaling the loss of various classes according to the inverse of the effective number of samples in each class, formulated as:

$$\ell_{\text{CB}} = -\sum_{i=1}^{|\mathcal{V}_{\text{L}}|} \frac{1-\beta}{1-\beta^{|C_{y_i}|}} \log\left(\ell_{\text{ERM},i}\right), \quad (5)$$

where $\ell_{\text{ERM},i}$ is the empirical risk of each labeled sample and β is a hyperparameter. However, this direct approach may not always be the best solution because it does not consider the graph topology. To address this, ReNode [31] focuses on both the quantity and topology imbalances in nodes by examining the shift in node influence and adaptively adjusting the weight of labeled nodes according to their relative positions to class boundaries. Similarly, TAM [32] also utilizes topological information, comparing each node's connectivity pattern to the average pattern of its class and adaptively modifying the margin based on that.

B. Augmentation-Based Approaches

Augmentation-based approaches aim to enhance model training with extra information, boosting performance in imbalanced learning scenarios. This approach includes two techniques: *information augmentation* and *transfer learning*.

Information Augmentation (IA): IA introduces additional knowledge into model training. IA techniques includes generating informative augmented views through transformations $\mathcal{T}_{\text{IA}} = \mathcal{T}_{\text{IA}}(\mathcal{G}) = \{\tilde{G}_1, \tilde{G}_2, \ldots\}$. These augmented data provide complementary learning signals that help alleviate class imbalance. The overall training loss typically consists of two components:

$$\ell = \ell_{\text{ERM}}\left(\mathcal{F}(\mathcal{G}), \mathbf{Y}\right) + \lambda \ell_{\text{AUX}}(\mathcal{T}_{\text{IA}}), \quad (6)$$

where ℓ_{ERM} is the standard empirical risk, e.g., cross-entropy loss, loss on the original graphs, and ℓ_{AUX} is an auxiliary loss that depends on the augmented data $\mathcal{T}_{\text{IA}}$. For example, GraphMixup [33] enhances graph augmentation via semantic feature mixup and contextual edge mixup for local and global structure, and a reinforcement mechanism that adaptively sets the upsampling ratio for each minority class. However, this augmentation lacks effective supervision and discriminative power. Instead of augmenting samples directly, G ^{2} GNN [34] builds a graph-of-graphs using kernel similarity to gain global supervision from neighbors and local supervision from the graphs themselves. CM-GCL [35] exploits multi-modal data through a co-modality framework that generates contrastive pairs, and applies inter- and intra-modal graph contrastive losses for optimization.

Transfer Learning (TL): TL aims to apply knowledge from one domain (e.g., specific datasets or classes) to enhance model training in another. In graph imbalanced learning, two main TL approaches are majority-to-minority transfer and knowledge distillation. The former leverages patterns from majority classes to improve prediction on minority classes. For instance, SOLT-GNN [36] captures co-occurrence substructures from majority graphs at both node- and subgraph-levels and uses a relevance predictor to transfer them to minority graphs. RAHNet [37] introduces a retrieval-augmented branch that retrieves informative graphs to enrich minority class representations. As another direction, knowledge distillation enables knowledge transfer via teacher-student training. GNN-INCM [38] incorporates a distillation module focusing on hard samples using distribution and triplet alignment losses. LTE4G [39] employs two student models: one for majority nodes and another for minority nodes, each specialized in classifying their respective classes.

C. Module Improvement Approaches

Research in this area focuses on enhancing network modules in imbalanced learning, encompassing *representation learning*, *classifier training*, and *model ensemble*.

Representation Learning (RL): While contrastive learning is commonly used to enhance representation [147], [148], ImGCL [40] identifies a limitation in current graph contrastive methods' ability to discriminate imbalanced nodes. To address this, ImGCL introduces a node centrality-based progressively balanced sampling method to better preserve the intrinsic graph structure. But no extra supervision is introduced. INS-GNN [41] first employs self-supervised learning to pre-train the model, then uses self-training to assign pseudo-labels to unlabeled nodes. Besides self-supervised learning, GNN-CL [42] applies metric learning, focusing on distance-based losses to learn an embedding space with improved discrimination. Specifically, it proposes neighbor-based triplet loss to distinguish minority-class samples by adjusting node distances in the feature space.

Classifier Training (CT): Imbalance causes larger weight norms for majority classes, biasing the classifier toward dominant ones. For better training, RAHNet [37] jointly learns a balanced feature extractor and unbiased classifier via decoupling. RAHNet mitigates classifier bias by regularizing classifier weights while keeping the extractor fixed.

Model Ensemble (ME): ME typically employs a multi-expert learning framework to conduct graph learning from diverse perspectives, thereby enhancing model performance. A typical ME method takes the form:

$$\hat{y} = \sum_{i=1}^{K} w_i \cdot \mathcal{F}_i(\mathcal{G}_i), \quad \sum_{i=1}^{K} w_i = 1, \tag{7}$$

where each $\mathcal{F}_i$ denotes the prediction from expert i, trained on a specific view, and w_i is its gating weight. Diversity is introduced by varying inputs or architectures, and the weights w_i may be static or learned dynamically based on input. GraphDIVE [43] focuses on extracting varied representations at both node and graph levels. To ensure the diversity of experts, each expert receives a specific graph representation view, which forms the final prediction through aggregation. Building upon the multi-expert framework, CoMe [44] combines various expert models using dynamic gating functions, enhancing the overall diversity of the training network. Moreover, it performs knowledge distillation in a disentangled manner among experts, motivating them to learn extra knowledge from each other.

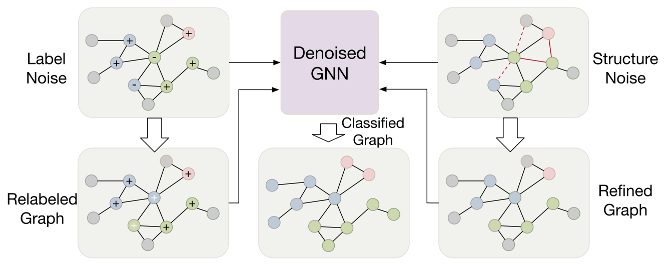

Fig. 4. Illustration of GNNs under the impact of label and structural noise. Inevitable label errors require the GNN model to accurately identify mislabeled samples, while the fake or absent edges between nodes require the model to reconstruct the ground-truth adjacency.

D. Discussion

Although significant advancements have been made in graph imbalanced learning, most work focuses on label imbalance, with less attention given to structural imbalances within graphs. Sun et al. [146] analyze supervision distribution globally, addressing under-reaching and over-squashing via position-aware structure learning that optimizes information propagation paths, addressing topology imbalance directly. Similarly, QTIAH-GNN [143] tackles topology imbalance with a multi-level, label-aware neighbor selection mechanism that aims to identify and sample neighbors similar to a given central node while effectively excluding nodes of dissimilar classes. However, deeper understanding and solutions for topology imbalance remain needed.

V. Noise

In addition to the challenge of class imbalance, the presence of data noise within the graph is a widespread issue in real-world scenarios [149]. There are three common types of noise: *label noise*, *structural noise* and *attribute noise*. Fig. 4 illustrates the fundamental principles of GNNs in addressing three noises. Next, we discuss these three aspects in detail.

A. Label Noise

This section introduces graph learning with label noise, covering both node-level [45], [47], [49], [50] and graph-level [150], [151], [152] classification. The former is more extensively studied, and our focus is on node-level classification. The goal is to train a GNN classifier $\mathcal{F}^*$ that is robust to label noise. The model is trained on a graph $\mathcal{G}$ where some nodes have noisy labels $\mathbf{Y}^{\text{noisy}}$, and outputs predicted labels $\mathbf{Y}_{\text{U}}$ for the unlabeled nodes as:

$$\mathbf{Y}_{\text{U}} = \mathcal{F}^*(\mathcal{G}), \ \mathcal{F}^* = \arg\min_{\mathcal{F}} \ell\left(\mathcal{F}(\mathcal{G}), \mathbf{Y}^{\text{noisy}}\right).$$

The expectation is that the trained model effectively mitigates the adverse impact of noisy labels on the predictions $\mathbf{Y}_{\text{U}}$ for the unlabeled nodes. Studies addressing scenarios with noisy labels on graphs can be categorized into two groups: *loss correction*

TABLE II
OVERVIEW OF METHODS ON GRAPHS AGAINST THREE TYPES OF NOISES. ("PLL": PSEUDO-LABEL LEARNING, "SR": SELF-REINFORCEMENT, "HR": HOMOPHILY RECONSTRUCTION, "ES": EDGE-LEVEL SAMPLING, "NS": NODE-LEVEL SAMPLING).

(a). Label Noise

Method	Data Type	Core Idea	Implementation & Details
NRGNN [45]	Node-level	LoC	Edge Connection, PLL
DND-Net [46]	Node-level	LoC	Decoupling, PLL
PIGNN [47]	Node-level	LoC	PI
CR-GNN [157]	Node-level	LoC	CL, Sample Selection
CP [48]	Node-level	LoC	Attack and Defense
RTGNN [49]	Node-level	LoC	SLP, SR
GraphCleaner [50]	Node-level	LoC	MTM, Neighbor Agreement
UnionNET [51]	Node-level	LaC	Label Aggregation
GNN Cleaner [52]	Node-level	LaC	LP
ERASE [53]	Node-level	LaC	CRR, LP
CGNN [55]	Node-level	LoC & LaC	CL, NV
LP4GLN [54]	Node-level	LaC	HR, LP
D-GNN [150]	Graph-level	LoC	MTM, Backward LoC
OMG [151]	Graph-level	LoC & LaC	Coupled Mixup, CL

(b). Structure Noise

Method	Type	Post-processing	Graph Regularization
GRCN [56]	KML	kNN	Sparsification
GNNGuard [57]	KML	ϵNN	Sparsification
GDC [58]	KML	kNN,ϵNN	Sparsification
GLCN [59]	NML	kNN	Sparsification
IDGL [60]	NML	ϵNN	Sparsification, Smoothness
SLAPS [61]	NML	kNN	Sparsification
DropEdge [62]	ES	ϵNN	Sparsification
DropCONN [63]	ES	-	Smoothness
FastGCN [65]	NS	-	-
PTDNet [64]	ES	-	Sparsification, Smoothness
NeuralSparse [66]	ES	kNN	Sparsification
TO-GCN [67]	DO	-	Smoothness
PRO-GNN [68]	DO	-	Sparsification, Smoothness
Xu et al. [160]	DO	-	Attack and Defense
Gosch et al. [69]	DO	-	Attack and Defense
PTA [70]	DO	-	Decoupling
RLP [71]	DO	-	Decoupling
PAMT [72]	DO	-	Decoupling

(c). Attribute Noise

Method	Type	Implementation & Details
Nettack [73]	Attack and Defense	Incremental Computations
BVAT [76]	Attack and Defense	Virtual Adversarial Perturbations
GCORN [77]	Defense	Orthonormal Weight Matrix
MQE [79]	Loss Refinement	Multi-hop Propagation
BRGCL [80]	Loss Refinement	Contrastive Learning

and *label correction*. Table II(a) presents the overview of these methodologies. In the subsequent sections, we delve into detailed introductions.

1) Loss Correction Approaches: Loss correction (LoC) approaches [45], [49] aim to rectify the impact of label noise on risk minimization by adapting the training loss. Various commonly employed techniques include *loss regularization*, *sample reweighting*, *adversarial attack and defense*, and *guidance from a mislabeled transition matrix*.

Loss Regularization: Loss regularization methods introduce additional information to alleviate the impact of noisy labels, where the total loss can be formulated as:

$$\ell = \ell_{\text{ERM}}\left(\mathcal{F}(\mathcal{G}), \mathbf{Y}^{\text{noisy}}\right) + \lambda \ell_{\text{REG}}, \tag{8}$$

where ℓ_{ERM} is the empirical risk and ℓ_{REG} is the regularization term with additional information. By incorporating such supplementary information, loss regularization enhances the model's robustness to noisy labels during training. For example, NRGNN [45] suggests connecting unlabeled nodes with labeled nodes by exploring high feature similarity, where the structure reconstruction loss is introduced to adjust the training loss. This approach also incorporates pseudo-label information with high prediction confidence to enhance supervision and further mitigate the impact of label noise. However, it overlooks the potential negative propagation caused by such connections. DND-Net [46] proposes a simple yet robust framework that decouples propagation and transformation to prevent noise spreading, integrating reliable pseudo-labeling with neighbor-aware uncertainty reweighting. From another perspective, PIGNN [47] focuses on pair-wise interaction (PI) between nodes to support noise-tolerant GNN learning, showing that PI introduces less noise than point-wise methods. Contrastive learning (CL) [153], [154], [155], [156] leverages comparisons between similar and dissimilar pairs to learn robust representations. CR-GNN [157] uses an unsupervised neighbor contrastive loss with a dynamic cross-entropy loss, selecting nodes with consistent predictions as reliable to prevent overfitting to noisy labels.

Sample Reweighting: Leveraging the memorization effect, training on small-loss samples is a promising approach to mitigate challenges posed by noisy labels [158]. In sample reweighting, the strategy down-weights large-loss samples in the training loss, thereby enhancing the supervision from clean labels. The corresponding loss can be unified into:

$$\ell = \sum_{i=1}^{|\mathcal{V}_\text{L}|} \omega_i \cdot \ell_{\text{ERM},i}, \quad \sum_{i=1}^{|\mathcal{V}_\text{L}|} w_i = 1, \tag{9}$$

where ω_i is the specific-defined weight and $\ell_{\text{ERM},i}$ is the empirical risk of each labeled node. Drawing inspiration from the edge predictor [45], RTGNN [49] integrates both sample reweighting and loss regularization techniques. Utilizing the small-loss principle (SLP), RTGNN filters out clean labels and diminishes the influence of noisy labels in the training process. Additionally, RTGNN introduces internal strengthening and coherence regularization as supplementary forms of supervision, aiming to enhance the model's robustness.

Adversarial Attack and Defense: Akin to noisy labels, adversarial label-flipping attacks strategically manipulate labels to mislead a model during training, deliberately introducing misclassifications for adversarial purposes. CP [48] develops an attack model LafAK, based on an approximated closed form of GNNs, to simulate label-flipping attacks. They also propose a defense framework that uses a self-supervised task to preserve community properties as regularization, thus improving robustness and mitigating overfitting.

Mislabeled Transition Matrix (MTM): MTM is instrumental in characterizing how nodes in different classes are mislabeled, effectively capturing the underlying pattern of noise formation. By utilizing this matrix, it guides the training process when dealing with noisy labels. GraphCleaner [50] first leverages the validation set to learn MTM and then uses the estimated MTM as a synthetic mislabel dataset generator to train the noise detector.

2) Label Correction Approaches: Label correction (LaC) approaches [52], [55] offer a more intuitive solution by identifying nodes with potentially incorrect labels and correcting them to ensure reliable training. The detect-and-correct procedure can be formulated as:

$$\mathbf{Y}^{\text{correct}} = \mathcal{C}_{\text{N}}\left(\mathcal{G}, \mathbf{Y}^{\text{noise}}\right), \ \mathbf{Y}^{\text{noise}} = \mathcal{D}_{\text{N}}\left(\mathcal{G}, \mathbf{Y}^{\text{noisy}}\right), \quad (10)$$

where $\mathcal{C}_{\text{N}}$ and $\mathcal{D}_{\text{N}}$ are tailored noise corrector and detector, respectively. Then the training loss is built upon the corrected $\mathbf{Y}^{\text{correct}}$. Common correction techniques include *label propagation*, *neighbor voting*, et al. These techniques contribute to refining the accuracy of labels in the training data, ultimately enhancing the robustness of the model against the impact of noisy labels during the learning process.

Label Propagation (LP): In the context of the homophily assumption, LP disseminates clean labels through the graph by leveraging node similarity, effectively correcting potentially mislabeled nodes. For example, UnionNET [51] uses similarities of learned node representations as attention weights for label aggregation, then employs aggregated class probabilities to weight training samples and guide label correction, jointly optimizing the network. However, it fails to distinguish the reliability of samples. GNN Cleaner [52] propagates clean labels through graph structure, generating pseudo-labels filtered by agreement with given labels, and then applies a learnable correction scheme supervised by reliable labels. Coding rate reduction (CRR) [159] promotes semantically rich representations, extended by ERASE [53] via decoupled propagation that combines placeholder and refined labels with fault-tolerant adjustments. Additionally, LP4GLN [54] tackles noisy labels in heterogeneous graphs by reconstructing homophily-restored graphs, iteratively selecting high-confidence labels through LP.

Neighbor Voting (NV): NV determines the corrected label for a node by considering the majority label among its neighbors. This approach assumes neighboring nodes are likely to be similar, emphasizing the role of local structure in correcting noisy labels. CGNN [55] combines Loc and Lac, employing graph CL as a regularization to avoid overfitting. It filters noisy labels based on consistency between predicted/annotated labels and neighbors, then corrects filtered noisy nodes via neighbor voting.

B. Structure Noise

Structure noise in GNNs refers to the presence of irrelevant or noisy information in the graph structure that can negatively impact the performance of the GNN model [161]. GNNs are highly susceptible to structure noise since errors can propagate throughout the graph due to the message-passing mechanism [10]. Therefore, the quality of the input graph structure is critical to achieving optimal GNN performance [162]. The mainstream methods of tackling structure noise are *graph structure learning* and *direct optimization*. The former focuses on optimizing the graph structure before carrying out downstream tasks and can be further categorized into *metric learning* and *sampling-based* approaches. They refine the noisy adjacency matrix and node representations by:

$$\mathbf{A}^* = \mathcal{F}_{\text{ADJ}}\left(\mathbf{A}^{\text{noisy}}, \mathbf{X}\right), \ \mathbf{Z}^* = \mathcal{F}_{\text{REP}}\left(\mathbf{X}, \mathbf{A}^*\right), \quad (11)$$

where $\mathcal{F}_{\text{ADJ}}$ and $\mathcal{F}_{\text{REP}}$ are the structure learner and representation learner, respectively. The latter mitigates the impact of structural noise by directly incorporating tailored regularization terms (ℓ_{REG}), improving the GNN architecture ($\bar{\mathcal{F}}$), or refining the supervision labels ($\bar{\mathbf{Y}}$) to optimize the training loss, where the loss can be unified as:

$$\ell = \ell_{\text{ERM}}\left(\bar{\mathcal{F}}\left(\mathbf{A}^{\text{noisy}}, \mathbf{X}\right), \bar{\mathbf{Y}}\right) + \lambda \ell_{\text{REG}}. \quad (12)$$

The following sections provide a comprehensive overview of these methods, which are summarized in Table II(b).

1) Metric Learning Approaches: Metric learning (ML) approaches treat the metric function as learnable parameters and refine the graph structures by learning the metric function $\phi(\cdot,\cdot)$ of pair-wise representations: $\tilde{\mathbf{A}}_{ij} = \phi(\mathbf{h}_i, \mathbf{h}_j)$, where $\mathbf{h}_i, \mathbf{h}_j$ are the learned embedding representations of nodes v_i, v_j, and $\tilde{\mathbf{A}}_{ij}$ denotes the learned edge weight between v_i and v_j. The refined matrix $\mathbf{A}^*$ is obtained as the output of an update function $g(\cdot,\cdot)$: $\mathbf{A}^* = g(\mathbf{A}^{\text{noisy}}, \tilde{\mathbf{A}})$. According to the different realizations of the metric function $\phi(\cdot,\cdot)$, ML approaches can be divided into *kernel-based* and *neural-based MLs*. Additionally, kNN (i.e., each node has up to k neighbors) and ϵNN (i.e., edges whose weight are less than ϵ will be removed) are two common post-processing operations to achieve graph sparsification.

Kernel-based ML (KML): This kind of method uses the kernel function as ϕ to calculate edge weights between nodes. GRCN [56] contains a GCN that predicts missing edges and revises edge weights based on node embeddings. It uses the dot product as a kernel function to calculate the similarity between each node. Yet it cannot achieve reasonable graph sparsification. GNNGuard [57] protects GNNs from adversarial attacks by detecting and removing suspicious edges, ensuring robust predictions. It uses cosine similarity to assess connection relevance. Graph diffusion convolution (GDC) [58] uses generalized graph diffusion for graph sparsification and improving learning outcomes, allowing for information aggregation from a broader neighborhood. GDC uses a diffusion kernel function to quantify edge connections:

$$\tilde{\mathbf{A}} = \sum_{k=0}^{\infty} \theta_k \mathbf{T}^k, \quad (13)$$

with the generalized transition matrix $\mathbf{T}$ and the weighting coefficients θ_k satisfying $\sum_{k=0}^{\infty} \theta_k = 1$. Note that $\mathbf{T}$ can be the random walk transition matrix $\mathbf{T}_{\text{rw}} = \mathbf{A}^{\text{noisy}}\mathbf{D}^{-1}$ and the symmetric transition matrix $\mathbf{T}_{\text{sym}} = \mathbf{D}^{-1/2}\mathbf{A}^{\text{noisy}}\mathbf{D}^{-1/2}$, where $\mathbf{D}$ is the diagonal matrix of node degrees.

Neural-based ML (NML): Compared to kernel-based approaches, neural-based approaches use more complex neural networks as the metric function $\phi(\cdot,\cdot)$ to calculate edge weights between nodes and learn an optimized graph structure. GLCN [59] seeks to improve the performance of GCN in semi-supervised learning tasks by learning an optimal graph structure. It utilizes a graph learning layer to calculate the similarity between two nodes and generates an optimal adaptive graph representation $\tilde{\mathbf{A}}$ for subsequent convolution operation. Formally, it learns a graph $\tilde{\mathbf{A}}$ as:

$$\tilde{\mathbf{A}}_{ij} = \frac{\exp(\mathrm{ReLU}(\boldsymbol{\alpha}^\top |\mathbf{z}_i - \mathbf{z}_j|))}{\sum_{j=1}^{|\mathcal{V}|} \exp(\mathrm{ReLU}(\boldsymbol{\alpha}^\top |\mathbf{z}_i - \mathbf{z}_j|))}, \tag{14}$$

where $\boldsymbol{\alpha}$ is the learnable parameter vector. IDGL [60] iteratively refines graph structure and GNN parameters to improve node embeddings and prediction accuracy. It uses weighted cosine similarity to optimize graph structure:

$$\tilde{\mathbf{A}}_{ij} = \frac{1}{m}\sum_{p=1}^{m} \tilde{\mathbf{A}}_{ij}^{p}, \quad \tilde{\mathbf{A}}_{ij}^{p} = \cos\left(\mathbf{w}_p \odot \mathbf{z}_i, \mathbf{w}_p \odot \mathbf{z}_j\right), \tag{15}$$

where $\odot$ denotes the hadamard product. Specifically, $\mathbf{w}$ is a learnable weight metric with m perspectives, and IDGL calculates the weighted average of cosine similarity for each head. SLAPS [61] employs complex neural networks as a metric function to learn task-specific graph structures via self-supervision. Its graph generator infers structures with learnable parameters, while denoising autoencoders enhance supervision through feature denoising.

2) Sampling-Based Approaches: Sampling-based approaches involve randomly sampling edges or nodes from the original input graph according to a specific ability distribution to generate a refined graph structure, formulated as:

$$\mathbf{A}^* = \mathcal{S}_{\mathrm{SAM}}\left(\mathbf{A}^{\mathrm{noisy}}, \mathbf{X}\right), \tag{16}$$

where $\mathcal{S}_{\mathrm{SAM}}(\cdot,\cdot)$ is designed based on the graph data itself or the specific task. This method allows for partial and random subset aggregation during GNN training, alleviating the structure noise and enhancing the model's robustness. Additionally, sampling approaches can be further categorized based on their relevance to downstream tasks.

Task-independent Approaches: This method involves sampling or dropping without considering their relation to downstream tasks. DropEdge [62] is an edge-level sampling technique that improves GCNs by randomly removing a certain portion of edges from the input graph during training. This technique acts as a form of unbiased data augmentation and reduces the intensity of message passing between nodes. However, However, the absence of targeted supervision limits its adaptability to specific downstream tasks. Instead, DropCONN [63] is a biased graph-sampling technique that aims to mitigate the effects of graph adversarial attacks. It penalizes adversarial edge manipulations by constructing random and deformed subgraphs, introducing a significant regularization effect on graph learning.

Task-dependent Approaches: Instead of directly deforming the graph structure, task-dependent approaches seek feedback from downstream tasks to make improvements. PTDNet [64] is an edge-level method that denoises graphs and enhances generalization by dropping task-irrelevant edges via a parameterized network. It also employs nuclear norm regularization to enforce a low-rank constraint on the graph. FastGCN [65] is a node-level approach that improves GCN training efficiency by sampling vertices based on importance, mitigating recursive neighborhood expansion without sacrificing accuracy. Similarly, NeuralSparse [66] removes task-irrelevant edges using a deep neural network based on structural and non-structural information.

3) Direct Optimization Approaches: Direct optimization (DO) approaches consider the adjacency matrix learnable, which are optimized by applying specific regularization or optimization methods, including *smoothness*, *adversarial attack and defense*, and *decoupling-based approaches*

Smoothness: This method is based on a commonly held assumption that graph signals change smoothly between adjacent nodes [163]. This assumption typically refers to the smoothness of features and labels, assuming that nearby and connected nodes are likely to share the same labels or similar node features. TO-GCN [67] uses label smoothness regularization, jointly and alternately optimizing network topology and updating GCN parameters by fully utilizing the label and topology information. Pro-GNN [68] uses feature smoothness regularization to recover a clean graph structure, jointly updating GNN parameters assisted by other techniques like low rank and sparsification.

Adversarial Attack and Defense: This method handles structure noise by adding adversarial changes to graph edges [164]. Xu et al. [160] introduces a gradient-based approach to help GNNs resist both greedy and gradient attacks without lowering accuracy. Gosch et al. [69] leverages learnable graph diffusion to adaptively defend against structural perturbations while satisfying global and local constraints.

Decoupling: Decoupling-based methods decoupled the GNN architecture and refine the conventional cross-entropy loss. Decoupled GCN has been theoretically shown to be equivalent to label propagation [70], and is notably robust to structure noise. To overcome its sensitivity to initialization and label noise, PTA [70] combines graph structural proximity and predictive confidence to dynamically reweight pseudo-labels in the cross-entropy loss, addressing both structure and label noise. To further integrate attribute information, RLP [71] adjusts the propagation matrix by combining attribute similarity with structural cues, and employs a momentum strategy for training stability. PAMT [72] uses a Hadamard product between an adaptive similarity mask and the adjacency matrix to mitigate inaccurate initial propagation caused by structure noise.

C. Attribute Noise

Attribute noise refers to errors or disruptions in the features of nodes or edges in a graph. These perturbations can harm model performance by distorting the original feature distributions. Such noise often comes from data collection mistakes, missing values, or even intentional adversarial attacks [164] that aim to corrupt the input features. The aim is to train a robust GNN $\mathcal{F}^*$ classifier that minimizes the impact of attribute perturbations δ:

$$\mathcal{F}^* = \arg\min_{\mathcal{F}} \ \ell\left(\mathcal{F}(\mathbf{X}^{\mathrm{noisy}}, \mathbf{A}), \mathbf{Y}\right). \tag{17}$$

Fundamental strategies against this noise fall into two categories: *adversarial attack and defense* and *loss refinement*. Table II(c) presents the overview of these methodologies.

1) Adversarial Attack and Defense Approaches: Adversarial attack and defense methods harden models against malicious perturbations through min-max optimization, formalized as:

$$\min_{\mathcal{F}} \max_{\|\delta\| \leq \epsilon} \ell\left(\mathcal{F}\left(\mathbf{X}+\delta, \mathbf{A}\right), \mathbf{Y}\right), \quad (18)$$

where δ denotes adversarial noise constrained by ϵ. For this type, several methods have been proposed to enhance the robustness of GNNs. For example, Nettack [73] is one of the first to reveal the vulnerability of GNNs. It generates subtle adversarial perturbations on both node features and graph structure. GraphAT [74] introduces a dynamic regularizer. It helps stop perturbations from spreading too far in the graph. GCNVAT [75] looks at sensitive feature directions. It smooths GCNs along these directions to reduce the effect of adversarial changes. BVAT [76] uses virtual adversarial training. It runs the training in batches to better learn local graph structures. GCORN [77] enhances inherent robustness against node feature attacks by enforcing orthonormal weight matrices, yielding an attack-agnostic robust GNN.

2) Loss Refinement Approaches: Loss refinement techniques enhance robustness by modifying the objective function, typically through regularization terms grounded in stability analysis:

$$\min_{\mathcal{F}} \ell_{\text{ERM}}\left(\mathcal{F}\left(\mathbf{X}^{\text{noisy}}, \mathbf{A}\right), \mathbf{Y}\right) + \lambda \ell_{\text{REG}}, \quad (19)$$

where ℓ_{REG} penalizes sensitivity to input perturbations (e.g., via Lipschitz constraints [165]). T2-GNN [78] proposes a dual teacher-student framework, where one teacher focuses on node features and the other on graph structure. Both teachers transfer clean patterns to assist in recovering corrupted attributes. MQE [79] takes a probabilistic view and learns noise-invariant meta-representations by estimating the quality of multi-hop features using a Gaussian model. Beyond explicit repair, BRGCL [80] offers a different angle, embedding Bayesian nonparametrics into contrastive learning. The model iteratively distills robust prototypes from nodes with high confidence.

D. Discussion

Noise in real world GNNs mainly includes label noise, structure noise and attribute noise, all of which degrade model performance and robustness. Beyond node-level robust GNN training against label noise, some studies focus on graph-level learning, aiming to achieve reliable predictions for unlabeled graphs in the presence of noisy labels, such as D-GNN [150] and OMG [151]. Methods addressing structure noise include meta-learning [166], adversarial attack [57], [68], [167], graph revision [56], [59], graph sampling [62], [63], and others [168], [169]. The main idea is to optimize the graph structure and enhance robustness. To understand model robustness (and inversely sensitivity), Lipschitz stability [165] serves as a key metric for quantifying sensitivity to structure noise, influenced by both perturbation magnitude and Lipschitz constant. It provides theoretical guidance for designing graph models robust to structural noise. Such analysis can also guide the theoretical development of structure-resilient graph models. Despite these efforts, work that directly targets attribute noise is still scarce. Most existing research focuses more on label noise and structural noise, highlighting a critical gap in GNN literatures for real world.

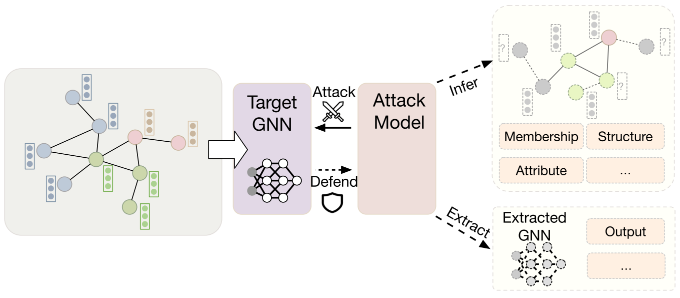

Fig. 5. Illustration of the attacks and defenses around both private data and model weights. The objective of the attack model is to extract private information from a target GNN. In response, the model needs to take measures and safeguard privacy from the attack model.

TABLE III
OVERVIEW OF METHODS FOR PRIVACY ATTACK/DEFEND ON GNNS MODELS. ("SR": STRUCTURE RECONSTRUCTION, "AR": ATTRIBUTE RECONSTRUCTION, "GPI": GENERAL PRIVACY ISSUES)

Method	Attack or Defend	Focused Problem	Implementation & Details
He et al. [82]	Attack	MIA	Node-level, Black-box
He et al. [171]	Attack	MIA	Edge-level, Black-box
Wu et al. [172]	Attack	MIA	Graph-level, Black-box
Duddu et al. [83]	Attack	AIA	Node-level, Black-box
GraphMI [84]	Attack	RA	White-box, SR
Defazio et al. [173]	Attack	MEA	Adversarial Framework
Wu et al. [85]	Attack	MEA	Information Leakage
Shen et al. [86]	Attack	MEA	Node-level, Black-box, Attack API access
DPNE [87]	Defend	MIA	DP
PrivGnn [88]	Defend	MIA	DP
DP-GNN [89]	Defend	MIA	DP
KProp [90]	Defend	MIA	DP
GERAI [91]	Defend	MIA	DP
DP-GCN [94]	Defend	AIA	LFD
DGCF [174]	Defend	AIA	LFD
GAL [92]	Defend	AR	AT
APGE [93]	Defend	AR	AT & LFD
SpreadGNN [95]	Defend	GPI	FL
D-FedGNN [96]	Defend	GPI	FL
GraphErase [97]	Defend	GPI	Machine Unlearning
watermark [175], [176]	Defend	MEA	MOV
MIAGraph [81]	Attack & Defend	MIA	DP

VI. Privacy

GNNs perform well on relational data and are widely used, but often overlook privacy risks in sensitive domains such as finance, e-commerce, and healthcare [18], [127]. Like other deep models, most GNNs are vulnerable to privacy attacks [170]. Fig. 5 outlines a general GNN privacy framework, and Table III summarizes related works.

A. Privacy Attack

Privacy attacks on GNNs target sensitive information – such as training data, node/link attributes, or model parameters –and are typically classified into four categories based on their objectives [18], [127].

Membership Inference Attack (MIA): MIAs aim to reveal whether a sample – such as a node [82], edge [171], subgraph [81], or entire graph [172] – is included in the training dataset, leading to potential information leakage [177]. Formally, with the GNN model $\mathcal{F}$ trained on training dataset $\mathcal{D}$, the attacker builds a binary classifier $\mathcal{A}_{\text{MIA}}$ such that:

$$\mathcal{A}_{\text{MIA}}\left(\mathcal{F}(v)\right) = \begin{cases} 1 & \text{if } v \in \mathcal{D}, \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

For instance, MIAGraph [81] trains a shadow model on data similar to the target model's training set and uses it to guide the attacker. The success of MIA largely relies on overfitting and information leakage from GNN, thus lacking consideration of the generalization gap.

Attribute (/Property) Inference Attack (AIA): AIAs aim to uncover data attributes that are not explicitly included in the feature set [83], [178]. Given a trained GNN model $\mathcal{F}$, a node $v_i \in \mathcal{D}$ with attributes $\mathbf{x}_i$, and its neighborhood $\mathcal{N}(v_i)$, There are some public attributes $\mathbf{x}_i^{\text{pub}} \subset \mathbf{x}_i$ used as features, and some private attributes excluded from input features $\mathbf{x}_i^{\text{priv}} = \mathbf{x}_i \backslash \mathbf{x}_i^{\text{pub}}$. The attacker's goal is to build an inference model $\mathcal{A}_{\text{AIA}}$ such that

$$\hat{\mathbf{x}}_i^{\text{priv}} = \mathcal{A}_{\text{AIA}}\left(\mathcal{F}, \mathbf{x}_i^{\text{pub}}, \{\mathbf{x}_j\}_{v_j \in \mathcal{N}(v_i)}\right) \tag{21}$$

can be used to estimate the private attributes of node v_i. If the attack is white-box, the attacker may also access the internal layers of $\mathcal{F}$, such as $\mathbf{h}_i^{(l)}$, a hidden representation of node v_i at layer l, and take the form of $\hat{\mathbf{x}}_i^{\text{priv}} = \mathcal{A}_{\text{AIA}}(\mathbf{h}_i^{(l)})$. When the graph structural information is well-protected and thus $\mathcal{N}(v_i)$ is unknown, attackers may learn to extract some global properties of the training graphs (e.g., average node degree, graph density, connectivity). Real-world datasets of all kinds can be targeted – e.g., GNNs on molecules (revealing chemical bonds, atom types) or social networks (leaking private traits like gender, age) [83].

Reconstruction (/Model Inversion) Attack (RA): RAs aim to infer private information of target samples and are generally categorized into attribute and structure reconstruction [18]. Attribute RAs recover node features $\mathbf{x}_i$ from embedding $\mathbf{h}_i$ via $\hat{\mathbf{x}}_i = \mathcal{A}_{\text{ARA}}(\mathbf{h}_i)$, while structure RAs infer graph topology, e.g., $\hat{\mathbf{A}}_{ij} = \mathcal{A}_{\text{SRA}}(\mathbf{h}_i^\top \mathbf{W} \mathbf{h}_j)$, where $\mathcal{A}_{\text{SRA}}$ can be a sigmoid function to get edge probabilities. Unlike AIAs, RAs target public attributes $\mathbf{x}_i^{\text{pub}}$ embedded in features [18]. These attacks often assume access to node embeddings [83], [84], [179], [180]. For instance, GraphMI [84] employs projected gradient descent and a graph auto-encoder to reconstruct the adjacency matrix, and feature explanations can further enhance structure inference [181].

Model Extraction (/Stealing) Attack (MEA): MEAs pose a significant threat to large models accessed via APIs [182], potentially enabling other privacy and adversarial attacks [18]. Attackers aim to replicate the target model by training a surrogate that mimics its performance and decision boundaries. Let $\mathcal{Q} = \{\mathcal{G}_i\}_{i=1}^{|\mathcal{Q}|}$ be a set of query graphs, and $\mathcal{F}(\mathcal{G}_i)$ the predictions returned by the model. The attacker's goal is to train a surrogate model $\hat{\mathcal{F}}$ such that:

$$\hat{\mathcal{F}}(\mathcal{G}_i) \approx \mathcal{F}(\mathcal{G}_i), \quad \forall \mathcal{G}_i \in \mathcal{Q}. \tag{22}$$

Some privacy attacks operate in a transductive or white-box setting, while others follow an inductive or black-box paradigm [183]. Early methods using adversarial frameworks achieved up to 80% output similarity [173], whereas more recent approaches report fidelity as high as 90% on transductive GNNs [85]. MEAs on inductive GNNs have also shown strong effectiveness, even when attackers are restricted to remote API access to the victim models [86].

B. Privacy Preservation

On the other side, various methods have been proposed to make GNNs more resilient to privacy attacks [127].

Differential Privacy (DP): DP offers formal privacy guarantees for both i.i.d. and graph data. It ensures that an algorithm's output remains nearly unchanged when applied to neighboring datasets $\mathcal{D}'$ differing from original dataset $\mathcal{D}$ by only a few records. Formally, a randomized algorithm $\mathcal{M}$ satisfies (ε, δ)-DF, if for all measurable subsets S of outputs:

$$\mathbb{P}\left[\mathcal{M}(\mathcal{D}) \in S\right] \leq e^{\varepsilon} \cdot \mathbb{P}\left[\mathcal{M}(\mathcal{D}') \in S\right] + \delta. \tag{23}$$

DP mitigates MIA by limiting individual data influence through noise injection, with strong theoretical guarantees [184]. DPNE [87] enforces DP in network embedding via perturbed matrix factorization that implicitly preserves DeepWalk/LINE properties, though suffers significant utility loss ($\sim$30% accuracy drop at $\epsilon = 1$) due to high gradient sensitivity in random walks. MIAGraph [81] combines output perturbation and homophily reduction. PrivGNN [88] trains a DP-protected teacher model on poisoned data, then distills it into a student model. DP-GNN [89] uses DP-SGD to privatize gradient updates. KProp [90] adds noise to node features pre-aggregation, relying on averaging to maintain utility. GERAI [91] addresses membership privacy in recommendation systems via dual-stage encryption, perturbing user features while optimizing a modified loss.

Latent Factor Disentangling (LFD): In typical GNNs, embeddings encode both sensitive and task-relevant information [93]. LFD addresses this by decomposing node embedding $\mathbf{h}_i$ into private and task-related components: $\mathbf{h}_i = [\mathbf{h}_i^{\text{priv}} || \mathbf{h}_i^{\text{pub}}]$, where $\mathbf{h}_i^{\text{priv}}$ captures private (sensitive) features, and $\mathbf{h}_i^{\text{pub}}$ captures utility (non-sensitive or public) features. Disentanglement enforces independence between them by minimizing mutual information, exactly corresponding to processing AIA. APGE [93], built on a graph autoencoder (GAE), augments the decoder with privacy-related labels to encourage label-invariant embeddings. In node-dependent privacy settings, DP-GCN [94] introduces a two-module framework: one disentangles sensitive and non-sensitive representations, and the other trains a GCN on non-sensitive components for downstream tasks. DGCF [174], originally for graph collaborative filtering, also yields privacy-preserving embeddings by separating user intentions.

Adversarial Training (AT): A common defense approach is to deliberately reduce the effectiveness of specific privacy attacks. This strategy is known as AT [127] or adversarial privacy-preserving [18]. This approach trains models to minimize attack success while preserving downstream task performance, typically framed as a min-max optimization where a GNN encoder $\mathcal{F}$ learns embeddings that are both task-relevant and privacy-preserving:

$$\min_{\mathcal{F}} \max_{\mathcal{S}} \ell_{\text{ERM}}(\mathcal{F}(\mathcal{G}), \mathbf{Y}) - \lambda \ell_{\text{ADV}} \left(\mathcal{S}(\mathcal{F}(\mathcal{G})), \mathbf{Y}\right), \tag{24}$$

where ℓ_{ERM} is the empirical risk for a downstream task, $\mathcal{S}$ is an adversary model to infer private information from embeddings, ℓ_{ADV} is the adversary's loss (e.g., predicting sensitive attributes), and λ is a hyper-parameter controlling the utility-privacy trade-off. GAL [92] defends against worst-case attackers via graph information obfuscation. APGE [93], framed as a LFD method, also uses AT via an adversarial autoencoder to learn privacy-preserving embeddings. NetFense [185] instead perturbs the graph structure (e.g., the adjacency matrix) to mislead attackers without adversarial model updates. Thus, AT is capable of addressing various adversarial modes, such as AIA, RA, MEA, etc.

Federated Learning (FL): FL enables collaborative model training across distributed clients without sharing raw data, preserving privacy by aggregating local updates for global model optimization [128], [186]. Mathematically, let $\mathcal{K}$ be the set of participating clients, $\mathcal{D}$ be the full dataset, $\mathbf{w}$ be the global model parameters, $\mathbf{w}_k$ be the local model parameters on client $k \in \mathcal{K}$, $\ell_k(\mathbf{w}_k)$ be the loss on client k over their private dataset $\mathcal{D}_k$, the federated objective is typically:

$$\min_{\mathbf{w}} \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \cdot \ell_k(\mathbf{w}), \tag{25}$$

where in each training round t, the federated averaging (FedAvg) algorithm proceeds as:

$$\begin{aligned} &\mathbf{w}_k^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \nabla \ell_k(\mathbf{w}^{(t)}), && \text{(local update)} \\ &\mathbf{w}^{(t+1)} \leftarrow \sum_{k \in \mathcal{K}} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \cdot \mathbf{w}_k^{(t+1)}. && \text{(global aggregation)} \end{aligned} \tag{26}$$

This framework ensures that only the local server accesses raw data, preventing potential information leakage. So it performs well in dealing with MIA and even more general privacy issues. Graph FL models are typically categorized into three types based on graph data distribution: (i) Inter-graph FL, where each client holds a subset of graph samples [187]; (ii) Intra-graph FL, where each client owns a subgraph [188]; (iii) Decentralized FL, where clients communicate directly and aggregate without a central server [189]. FL faces challenges such as training with partial labels in decentralized settings. SpreadGNN [95] addresses this with DPA-SGD, while D-FedGNN [96] adopts DP-SGD for a similar solution.

C. Discussion

In addition to the methods discussed, other privacy-preserving techniques have been developed, such as machine unlearning [97], where GraphErase [97] uses balanced graph partitioning to maintain performance after node removal. Model-ownership verification (MOV) [175], [176] embeds watermarks into models to protect intellectual property. As privacy concerns grow, especially for proprietary models or those trained on sensitive data, understanding and defending against privacy attacks is crucial. Several defense strategies for GNNs exist, each with trade-offs: DP adds noise for strong guarantees but may reduce utility; LFD separates sensitive from task-relevant features for better control but requires prior knowledge; AT simulates attacks to balance defense and accuracy but can be unstable; FL avoids data sharing to preserve privacy but remains vulnerable without enhancements like DP. These methods complement each other depending on threat models and use cases.

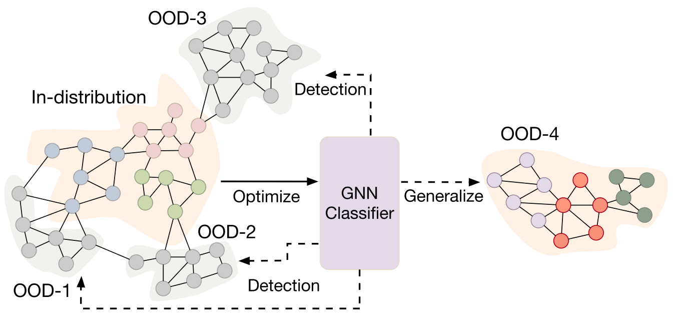

Fig. 6. Illustration of the OOD issue in real-world GNN training. While the model is trained on observed in-distribution data, the presence of OOD data calls for the development of mechanisms for OOD detection and generalization.

VII. Out-of-Distribution

Despite the strong representation capabilities of GNNs, they often display a mix of unsuitability and overconfidence when test sample distribution significantly deviates from the distribution of training samples. In this section, we delve into the Out-Of-Distribution (OOD) problem on graphs. There are two common OOD scenarios: *OOD detection* and *OOD generalization*. Fig. 6 illustrates the basic schematic diagrams for these two scenarios in GNNs. Next, we will discuss these two aspects in detail.

A. Out-of-Distribution Detection

OOD detection on graphs, which aims to distinguish test samples from the major in-distribution (ID) training data, has become an essential problem in real-world applications. Formally, we assume there is an ID graph dataset $\mathcal{D}^{\text{in}} = \{D_1^{\text{in}}, \cdots D_{N_1}^{\text{in}}\}$ and an OOD graph dataset $\mathcal{D}^{\text{out}} = \{D_1^{\text{out}}, \cdots D_{N_2}^{\text{out}}\}$, where data are sampled from a major distribution $\mathbb{P}^{\text{in}}$ and an OOD distribution $\mathbb{P}^{\text{out}}$, respectively. The general purpose of OOD detection on graphs is to identify its source distribution (i.e., $\mathbb{P}^{\text{in}}$ or $\mathbb{P}^{\text{out}}$) based on the learned detector $\mathcal{T}$:

$$\mathcal{T}(D_v; \tau, s, \mathcal{F}) = \begin{cases} D_v \in \mathbb{P}^{\text{in}}, & \text{if} \quad s(D_v, \mathcal{F}) \le \tau \\ D_v \in \mathbb{P}^{\text{out}}, & \text{if} \quad s(D_v, \mathcal{F}) > \tau \end{cases}, \tag{27}$$

where $\mathcal{F}$ is the trained model, s is a scoring function and τ is the corresponding threshold. D_v can be a node or a graph corresponding to the node-level or the graph-level task.

Based on the different scoring function designs, existing OOD detection methods on graphs can be roughly partitioned into: *propagation-based* approaches, *classification-based* approaches and *self-supervised learning-based* approaches. Table IV(a) presents the overview of these methods and we present a comprehensive introduction of these as follows.

TABLE IV
OVERVIEW OF METHODS FOR GRAPH OOD DETECTION AND GENERALIZATION. ("BAP": BAYESIAN POSTERIOR, "BLO": BI-LEVEL OPTIMIZATION, "SI": STATISTICAL INDEPENDENCE, "GNAS": GRAPH NEURAL ARCHITECTURE SEARCH)

(a). OOD Detection

Method	Task Type	Core Idea	Implementation & Details
GPN [98]	Node-level	Propagation	BaP, UE
GNNSage [99]	Node-level	Propagation	Energy, BeP
OODGAT [100]	Node-level	Propagation	Entropy Regularization
OSSNC [101]	Node-level	Propagation	VI, BLO
AAGOD [102]	Graph-level	Classification	Data-Centric, Amplifier
BWGNN [103]	Node-level	Classification	Graph Wavelet
GKDE [104]	Node-level	Classification	UE
iGAD [105]	Node-level	Classification	Graph Kernel
GLocalKD [106]	Graph-level	SSL	KD
GOOD-D [107]	Graph-level	SSL	CL
GRADATE [108]	Graph-level	SSL	CL
GLADC [109]	Graph-level	SSL	GAE, GR
GraphDE [110]	Node-level	SSL	VI, GR
OCGIN [111]	Graph-level	SSL	OCC
OCGTL [112]	Graph-level	SSL	OCC
GOODAT [107]	Graph-level	SSL	IB
SIGNET [191]	Graph-level	SSL	IB, Hypergraph
SGOOD [192]	Graph-level	SSL	Substructure

(b). OOD Generalization

Method	Task Type	Core Idea	Implementation & Details
DIR [193]	Graph-level	Subgraph	CI, Intervention
CAL [114]	Graph-level	Subgraph	CI, Disentanglement
CIGA [115]	Graph-level	Subgraph	CI, CL
StableGNN [116]	Graph-level	Subgraph	CI, Regularization
SizeShiftReg [117]	Graph-level	Subgraph	IL, Regularization
GIL [118]	Graph-level	Subgraph	IL, Regularization
FLOOD [194]	Node-level	Subgraph	IL, CL
LiSA [120]	Graph & Node	Subgraph	IL, Regularization
EERM [121]	Node-level	Subgraph	IL, AL
GraphAT [74]	Node-level	AL	Augmentation
CAP [122]	Node-level	AL	Augmentation
WT-AWP [125]	Node-level	AL	Augmentation, Regularization
LECI [124]	Graph-level	AL	IL, Augmentation
AIA [123]	Graph-level	AL	Augmentation
OOD-GNN [129]	Graph-level	RD	SI
GRACES [195]	Graph-level	RD	GNAS
OOD-LP [196]	Edge-level	TA	-

1) Propagation-Based Approaches: Unlike i.i.d. OOD detection in CV/NLP, node-level OOD detection involves connected ID and OOD nodes. Propagation-based methods adapt label propagation or GNN message passing to transfer uncertainty estimation (UE) frameworks. The process is typically represented as:

$$s_i^{(t)} = \alpha s_i^{(t-1)} + (1-\alpha) \sum_{v_j \in \mathcal{N}(v_i)} \Pi_{i,j} s_j^{(t-1)}, \tag{28}$$

where $\Pi_{i,j}$ reflects the importance of neighbor node v_j on v_i, α controls the concentration parameter of scoring. GPN [98] explores uncertainty quantification for OOD node detection. The method extends the input-dependent Bayesian update and explicitly models epistemic and aleatoric uncertainty by propagating node-wise estimates along the graph. However, Bayesian-based methods typically suffer from high computational complexity, which has motivated research into alternative approaches and approximate implementations. GNNSage [99] proposes a node-level OOD detection based on an energy function and introduces an energy-based belief propagation (BeP), which propagates the estimated energy score among nodes in the graph iteratively. OODGAT [100] explicitly models the interaction between ID and OOD nodes and separates these two types of nodes during feature propagation. OSSNC [101] learns to mix neighbors to mitigate the propagation to and from OOD nodes in a variational inference (VI) framework for simultaneous node classification and OOD detection.

2) Classification-Based Approaches: Another typical OOD detection method originated from a simple baseline, which uses the maximum softmax probability as the indicator scores of ID-ness [190]. The formation of the OOD score can be:

$$s\left(D_v, \mathcal{F}\right) = \mathrm{Max}\left(\mathcal{F}(D_v)\right). \tag{29}$$

AAGOD [102] proposes a data-centric post-hoc method instead of re-training the model for graph-level OOD detection. The method adopts a learnable amplifier generator to enlarge the indicator score gap between OOD and ID graphs. Some classification-based approaches also focus on node-level OOD detection and graph anomaly detection. BWGNN [103] uses the Beta wavelet kernel as a tailored spectral filter in GNN for node anomaly detection. GKDE [104] considers multidimensional uncertainties for node-level OOD detection. iGAD [105] treats graph-level anomaly detection as a special case of graph classification and proposes a dual-discriminative framework with GNN and graph kernel together to learn the label.

3) Self-Supervised Learning-Based Approaches: Since data labeling on graph-structured data is commonly time-consuming and labor-intensive [197], recent studies also consider the scarcity of class labels and OOD samples. The basic idea is to learn a self-supervised learning (SSL) framework for OOD detection on graphs based on unlabeled ID data and the method is mainly focused on graph-level OOD detection or anomaly detection.

Contrastive Learning (CL): A popular approach for self-supervised graph OOD detection is to drive multiple views of a graph sample and detect the OOD sample based on inconsistency. GLocalKD [106] jointly learns two GNNs and performs graph-level and node-level random knowledge distillation (KD) between the learned representations of two GNNS to learn a graph-level anomaly detector. But the singularity of perspective and scale limits its semantic richness. GOOD-D [107] performs perturbation-free graph data augmentation and utilizes hierarchical CL on the generated graphs for graph-level OOD detection. GRADATE [108] proposes a multi-view multi-scale CL framework with node-node, node-subgraph and subgraph-subgraph contrast for graph anomaly detection.

Graph Reconstruction (GR): Some works also aim to discriminative representations through reconstruction mechanisms and infer the graph OOD samples. GLADC [109] uses graph CL to learn node-level and graph-level representations and measure anomalous graphs with the error between generated reconstruction graph representations and original graph representations in a graph convolution autoencoder way. GraphDE [110] models the generative process of the graph to characterize the distribution shifts. Thus, ID and OOD graphs from different distributions indicate different environments and can be inferred by VI.

One-Class Classification (OCC): The goal of OCC is to train embeddings to cluster within a defined hypersphere, establishing a decision boundary. OCGIN [111] studies an end-to-end GNN model with OCC for anomaly detection. OCGTL [112] further extends the deep OCC method to a self-supervised detection way using neural transformations graph transformation learning as regularization. GOODAT [113] introduces a graph test-time OOD detection method under the graph information bottleneck (IB) principle to capture informative subgraphs. The surrogate labels are inherently ID, which can be seen as another kind of OCC.

B. Out-of-Distribution Generalization

Another key challenge in real-world OOD scenarios is graph OOD generalization [198], [199], addressing distribution shifts between training/test data. It covers node-level (node classification) and graph-level tasks (graph classification), with the latter being more studied. Let $\mathcal{F}$ denote the GNN classifier.The objective is to find the optimal $\mathcal{F}^*$ satisfying:

$$\mathcal{F}^* = \arg\min_{\mathcal{F}} \sup_{e\in E} \mathbb{E}_{(\mathcal{G},y)\in\mathcal{S}^e} \left[\ell\left(\mathcal{F}(\mathcal{G}), y\right)\right],$$

where E is a set to collect all the test environments, and $\mathcal{S}^e$ include all graph-label pairs in the environment e. $\ell(\mathcal{F}(\mathcal{G}), y)$ calculates the tailored loss for each sample. Distribution shift typically involves attributive shift (node attribute changes from varying backgrounds/environments) and structural shift (adjacency matrix variations from connectivity/graph size differences). Table IV(b) presents a overview of these methods and we introduce these below.

1) Subgraph-Based Approaches: Subgraph-based approaches [193] assume that every graph consists of a crucial part and a non-crucial part from semantic and environmental information, respectively. To identify subgraphs with crucial knowledge, they usually utilize causal inference and invariant theory for effective graph representation learning.

Causal Inference (CI): A key research direction constructs a structural causal graph (SCG) for theoretical analysis (TA), modeling interactions between invariant and spurious components. Following [193], invariant component is typically extracted via learnable subgraph masking, with loss functions enforcing invariance to non-causal features, which can result in a common form of the loss objectives as:

$$\min \ell_{\text{ERM}}\left(\mathcal{F}(\mathcal{G}), \mathbf{Y}\right) + \lambda\ell_{\text{VAR}}, \tag{30}$$

where ℓ_{ERM} denotes the empirical risk on the training dataset and ℓ_{VAR} is related to the variances of the predictions with different simulated spurious factors viewed as intervention. Based on this framework, numerous advanced variants are developed by integrating different techniques. For example, CAL [114] incorporates graph representations into the SCG, followed by the attention mechanism and representation disentanglement to select causal patterns. Besides, shortcut features are included in graph representations using the backdoor adjustment theory. CAL's attention-based causal selection lacks theoretical invariance guarantees and relies on predefined causal assumptions. Further, CIGA [115] considers the graph generation process with and without partial interaction between invariant and spurious parts, and then identifies the crucial subgraph by maximizing the intra-class semantics for invariance. In addition, StableGNN [116] adopts a differentiable graph pooling operator for subgraph extraction, which is optimized using a distinguishing regularizer with reduced spurious correlations.

Invariant Theory (IT): As in (30), causality-based methods relate IT to interventions. Other approaches leverage IT via data augmentation to improve robustness against distribution shifts by extending ERM to invariant risk minimization. For instance, SizeShiftReg [117] simulates size shifts with graph coarsening and applies a regularization loss for consistency. GIL [118] learns subgraph masks and uses invariance regularization. FLOOD [194] employs node dropping and attribute masking for contrastive bootstrapping [200]. MoleOOD [119] uses VI to guide invariant learning (IL). LiSA [120] generates variational subgraphs under information constraints and encourages diversity via energy-based regularization. EERM [121] simulates adversarial virtual environments for node-level IL.

2) Adversarial Learning Approaches: Adversarial learning (AL) has been widely utilized for OOD generalization [201] to reduce the domain discrepancy, which is naturally extended to graph data. Some approaches utilize AT to generate effective perturbation for enhancing the generalization capacity. For example, GraphAT [74] adds learnable perturbations to the target graph, which are trained to degrade the smoothness, addressing the worst-case problem. Yet it underutilizes the graph structure and environmental information. CAP [122] maximizes the training loss in the neighborhood of model parameters and node attributes, which can mitigate the risk of falling into the local minima. AIA [123] generates augmented data by merging dual masks for environmental/stable features, preserving semantics. Its regularization terms constrain perturbations to ensure optimization stability. LECI [124] adopts causal analysis from [115] to remove spurious correlations. It uses AT to enforce subgraph independence from labels/environments via an discriminator. WT-AWP [125] adapts the adversarial weight perturbation to graph classification as a regularization term, which is applied on partial layers to relieve potential gradient vanishing. Several domain adaption approaches also utilize AT to align graph representations across different domains [126], [202]. DEAL [126] leverages adversarial perturbation on node attributes to transfer source graphs into the target domain. In summary, these AT approaches can implicitly reduce the distribution discrepancy in the embedding space across domains. However, they usually require prior knowledge of domain or environment labels.

3) Discussion: Beyond designing scoring functions for OOD detection on graphs, recent efforts have emphasized explainability and generalization. For graph-level OOD and anomaly detection, methods such as SIGNET [191] jointly produce anomaly scores and explanatory subgraphs by maximizing mutual information across multi-view subgraphs, while SGOOD [192] leverages substructure information for enhanced representations. Meanwhile, generalization techniques like CL [115],

[194], [203], [204] and representation decorrelation (RD) [129], [195] have been adapted to improve OOD robustness. Theoretical studies also explore OOD link prediction [196], and large-scale benchmarks [198] have been developed to support empirical evaluation. These advances have been applied in real-world domains such as molecular property prediction and drug discovery [119].

VIII. Conclusion and Future Work

In summary, this survey presents a comprehensive review of how real-world GNNs tackle four major challenges: *imbalance, noise, privacy, and OOD*, which are often underrepresented in prior surveys. We first discuss the vulnerabilities and limitations of current models to reveal critical challenges, followed by a detailed categorization of existing methods addressing each aspect. Representative works are highlighted for their key contributions. In the following, we conclude with forward-looking discussions on promising future directions in real-world GNN research.

Enhancing Scalability: Most existing studies address imbalance, noise, privacy, and OOD challenges on small-scale graph datasets, leaving a gap with the large-scale graphs common in real-world scenarios. These issues become more complex at scale, requiring models with greater efficiency and robustness. For example, G^2GNN [34] alleviates imbalance by constructing a graph-of-graphs using graph kernel-based similarity, but its reliance on pairwise computations limits scalability. Exploring pre-training on small graphs and transferring to large, imbalanced, noisy, or OOD graphs remains a promising and impactful direction.

Improving Interpretability: Many real-world GNN applications, such as drug discovery, healthcare, and traffic planning, require high interpretability. Although existing methods have demonstrated strong performance under challenges like class imbalance and OOD generalization, interpretability remains underexplored. Enhancing model transparency through explanations is vital for reliability and robustness against attacks [205]. For example, SIGNET [191] jointly outputs anomaly scores and explanatory subgraphs via multi-view mutual information maximization. Incorporating built-in interpretability, post-hoc explanation, and counterfactual reasoning offers a promising path toward trustworthy GNNs in sensitive domains.

More Theoretical Guarantees: Establishing theoretical guarantees is essential for building reliable GNNs in real-world settings. While prior work has mainly focused on expressive power [206], theoretical insights into GNNs' generalization under noise, distribution shifts, and adversarial conditions remain limited. Such analyses can validate GNNs' robustness to natural perturbations and adversarial attacks, supporting their deployment in safety-critical applications. For example, GraphGuard [207] offers provable robustness against structural and feature perturbations in graph classification. Advancing theoretical guarantees is vital under scenarios like class imbalance and label noise, and developing unified analytical frameworks.

Comprehensive Benchmarks and Universal Models: Real-world scenarios are often studied in isolation, with existing models tailored to specific settings, limiting their generalizability. For example, UDA-GCN [202], designed for domain adaptation, fails under more complex tasks such as transfer learning with noisy labels. So, a comprehensive benchmark is needed to systematically assess models across diverse real-world challenges and yield an integrated score. Such a benchmark would facilitate fair evaluation and drive progress toward developing universally robust GNNs capable of performing well across heterogeneous conditions.

Towards More Realistic Applications: Building realistic GNN models is critical for broader deployment in domains such as biology, finance, and transportation. For instance, GNNs aid in analyzing protein-protein interaction networks [208], which are often imbalanced and exhibit OOD behavior when applied to new organisms. In finance, GNNs effectively detect rare fraudulent transactions in large-scale networks [142], while in transportation, they support route optimization under dynamic conditions [209]. These applications highlight the need for GNNs that can handle class imbalance, adapt to OOD data, and incorporate online learning to manage evolving environments.

References

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[2] W. Ju et al., "A comprehensive survey on deep graph representation learning," *Neural Netw.*, vol. 173, 2024, Art. no. 106207.

[3] Y. Xu, L. Zhu, J. Li, F. Li, and H. T. Shen, "Temporal social graph network hashing for efficient recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3541–3555, Jul. 2024.

[4] Q. Yao, Z. Shen, Y. Wang, and D. Dou, "Property-aware relation networks for few-shot molecular property prediction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 8, pp. 5413–5429, Aug. 2024.

[5] H. Li et al., "A survey on graph neural networks in intelligent transportation systems," 2024, *arXiv:2401.00713*.

[6] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018.

[8] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[9] W. Ju et al., "A survey of data-efficient graph learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2024, pp. 8104–8113.

[10] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1263–1272.

[11] R. Zhu et al., "Aligraph: A comprehensive graph neural network platform," in *Proc. VLDB Endowment*, vol. 12, no. 12, 2019, pp. 2094–2105.

[12] A. Pal, C. Eksombatchai, Y. Zhou, B. Zhao, C. Rosenberg, and J. Leskovec, "Pinnersage: Multi-modal user embedding framework for recommendations at pinterest," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 2311–2320.

[13] J. Yuan et al., "EGODE: An event-attended graph ode framework for modeling rigid dynamics," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, pp. 59093–59118.

[14] X. Luo et al., "HOPE: High-order graph ode for modeling interacting dynamics," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 23124–23139.

[15] M. Xu, M. Liu, W. Jin, S. Ji, J. Leskovec, and S. Ermon, "Graph and geometry generative modeling for drug discovery," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2023, pp. 5833–5834.

[16] W. Ju et al., "Few-shot molecular property prediction via hierarchically structured learning on relation graphs," *Neural Netw.*, vol. 163, pp. 122–131, 2023.

[17] H. Zhang, B. Wu, X. Yuan, S. Pan, H. Tong, and J. Pei, "Trustworthy graph neural networks: Aspects, methods, and trends," *Proc. IEEE*, vol. 112, no. 2, pp. 97–139, Feb. 2024.

[18] E. Dai et al., "A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability," *Mach. Intell. Res.*, vol. 21, no. 6, pp. 1011–1061, 2024.
[19] J. Choi, H. Kim, and J. J. Whang, "Unveiling the threat of fraud gangs to graph neural networks: Multi-target graph injection attacks against GNN-based fraud detectors," in *Proc. AAAI Conf. Artif. Intell.*, 2025, pp. 16028–16036.
[20] S. Li, H. Hua, and S. Chen, "Graph neural networks for single-cell omics data: A review of approaches and applications," *Brief. Bioinf.*, vol. 26, no. 2, 2025, Art. no. bbaf109.
[21] D. Lei, Z. Song, Y. Yuan, C. Li, and L. Zhu, "Achieving personalized privacy-preserving graph neural network via topology awareness," in *Proc. ACM Web Conf.*, 2025, pp. 3552–3560.
[22] X. Lin et al., "Conformal graph-level out-of-distribution detection with adaptive data augmentation," in *Proc. ACM Web Conf.*, 2025, pp. 4755–4765.
[23] B. Wu et al., "Trustworthy graph learning: Reliability, explainability, and privacy protection," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 4838–4839.
[24] L. Oneto et al., "Towards learning trustworthily, automatically, and with guarantees on graphs: An overview," *Neurocomputing*, vol. 493, pp. 217–243, 2022.
[25] J. Li et al., "Recent advances in reliable deep graph learning: Inherent noise, distribution shift, and adversarial attack," 2022, *arXiv:2202.07114*.
[26] T. Zhao, X. Zhang, and S. Wang, "Graphsmote: Imbalanced node classification on graphs with graph neural networks," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2021, pp. 833–841.
[27] L. Qu, H. Zhu, R. Zheng, Y. Shi, and H. Yin, "ImGAGN: Imbalanced network embedding via generative adversarial graph networks," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 1390–1398.
[28] J. Park, J. Song, and E. Yang, "Graphens: Neighbor-aware ego network synthesis for class-imbalanced node classification," in *Proc. Int. Conf. Learn. Representations*, 2022.
[29] X. Gao, W. Zhang, T. Chen, J. Yu, H. Q. V. Nguyen, and H. Yin, "Semantic-aware node synthesis for imbalanced heterogeneous information networks," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2023, pp. 545–555.
[30] W. Ju et al., "Cluster-guided contrastive class-imbalanced graph classification," in *Proc. AAAI Conf. Artif. Intell.*, 2025, pp. 11924–11932.
[31] D. Chen et al., "Topology-imbalance learning for semi-supervised node classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 29885–29897.
[32] J. Song, J. Park, and E. Yang, "TAM: Topology-aware margin loss for class-imbalanced node classification," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 20369–20383.
[33] L. Wu, J. Xia, Z. Gao, H. Lin, C. Tan, and S. Z. Li, "GraphMixup: Improving class-imbalanced node classification by reinforcement mixup and self-supervised context prediction," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2022, pp. 519–535.
[34] Y. Wang, Y. Zhao, N. Shah, and T. Derr, "Imbalanced graph classification via graph-of-graph neural networks," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2022, pp. 2067–2076.
[35] Y. Qian, C. Zhang, Y. Zhang, Q. Wen, Y. Ye, and C. Zhang, "Co-modality graph contrastive learning for imbalanced node classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 15862–15874.
[36] Z. Liu, Q. Mao, C. Liu, Y. Fang, and J. Sun, "On size-oriented long-tailed graph classification of graph neural networks," in *Proc. ACM Web Conf.*, 2022, pp. 1506–1516.
[37] Z. Mao, W. Ju, Y. Qin, X. Luo, and M. Zhang, "RAHNet: Retrieval augmented hybrid network for long-tailed graph classification," in *Proc. ACM Int. Conf. Multimedia*, 2023, pp. 3817–3826.
[38] Z. Huang, Y. Tang, and Y. Chen, "A graph neural network-based node classification model on class-imbalanced graph data," *Knowl.-Based Syst.*, vol. 244, 2022, Art. no. 108538.
[39] S. Yun, K. Kim, K. Yoon, and C. Park, "LTE4G: Long-tail experts for graph neural networks," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2022, pp. 2434–2443.
[40] L. Zeng, L. Li, Z. Gao, P. Zhao, and J. Li, "ImGCL: Revisiting graph contrastive learning on imbalanced node classification," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 11138–11146.
[41] X. Juan, F. Zhou, W. Wang, W. Jin, J. Tang, and X. Wang, "INS-GNN: Improving graph imbalance learning with self-supervision," *Inf. Sci.*, vol. 637, 2023, Art. no. 118935.
[42] X. Li et al., "Graph neural network with curriculum learning for imbalanced node classification," *Neurocomputing*, vol. 574, 2024, Art. no. 127229.
[43] F. Hu, W. Liping, L. Qiang, S. Wu, L. Wang, and T. Tan, "GraphDive: Graph classifcation by mixture of diverse experts," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 2059–2065.
[44] S. Yi et al., "Towards long-tailed recognition for graph classification via collaborative experts," *IEEE Trans. Big Data*, vol. 9, no. 6, pp. 1683–1696, Dec. 2023.
[45] E. Dai, C. Aggarwal, and S. Wang, "NRGNN: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2021, pp. 227–236.
[46] K. Ding, X. Ma, Y. Liu, and S. Pan, "Divide and denoise: Empowering simple models for robust semi-supervised node classification against label noise," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2024, pp. 574–584.
[47] X. Du et al., "Noise-robust graph learning by estimating and leveraging pairwise interactions," *Trans. Mach. Learn. Res.*, vol. 172, 2023, Art. no. 106113.
[48] M. Zhang, L. Hu, C. Shi, and X. Wang, "Adversarial label-flipping attack and defense for graph neural networks," in *Proc. IEEE Int. Conf. Data Mining*, 2020, pp. 791–800.
[49] S. Qian et al., "Robust training of graph neural networks via noise governance," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2023, pp. 607–615.
[50] Y. Li, M. Xiong, and B. Hooi, "GraphCleaner: Detecting mislabelled samples in popular graph learning benchmarks," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 20195–20209.
[51] Y. Li, J. Yin, and L. Chen, "Unified robust training for graph neural networks against label noise," in *Proc. Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2021, pp. 528–540.
[52] J. Xia et al., "GNN cleaner: Label cleaner for graph structured data," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 2, pp. 640–651, Feb. 2024.
[53] L.-H. Chen et al., "Erase: Error-resilient representation learning on graphs for label noise tolerance," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2024, pp. 270–280.
[54] Y. Cheng, C. Shan, Y. Shen, X. Li, S. Luo, and D. Li, "Label propagation for graph label noise," 2023, *arXiv:2310.16560*.
[55] J. Yuan, X. Luo, Y. Qin, Y. Zhao, W. Ju, and M. Zhang, "Learning on graphs under label noise," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2023, pp. 1–5.
[56] D. Yu, R. Zhang, Z. Jiang, Y. Wu, and Y. Yang, "Graph-revised convolutional network," in *Proc. Mach. Learn. Knowl. Discov. Databases: Eur. Conf.*, 2021, pp. 378–393.
[57] X. Zhang and M. Zitnik, "GNNGuard: Defending graph neural networks against adversarial attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 9263–9275.
[58] J. Gasteiger, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 13366–13378.
[59] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11313–11320.
[60] Y. Chen, L. Wu, and M. Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 19314–19326.
[61] B. Fatemi, L. El Asri, and S. M. Kazemi, "SLAPS: Self-supervision improves structure learning for graph neural networks," in *Proc . Adv. Neural Inf. Process. Syst.*, 2021, pp. 22667–22681.
[62] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards deep graph convolutional networks on node classification," in *Proc. Int. Conf. Learn. Representations*, 2020.
[63] L. Chen, X. Li, and D. Wu, "Enhancing robustness of graph convolutional networks via dropping graph connections," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2020, pp. 412–428.
[64] D. Luo et al., "Learning to drop: Robust graph neural network via topological denoising," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2021, pp. 779–787.
[65] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," in *Proc. Int. Conf. Learn. Representations*, 2018.
[66] C. Zheng et al., "Robust graph representation learning via neural sparsification," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 11458–11468.

[67] L. Yang, Z. Kang, X. Cao, D. Jin, B. Yang, and Y. Guo, "Topology optimization based graph convolutional network," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 4054–4061.
[68] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 66–74.
[69] L. Gosch, S. Geisler, D. Sturm, B. Charpentier, D. Zügner, and S. Günnemann, "Adversarial training for graph neural networks: Pitfalls, solutions, and new directions," in *Proc . Adv. Neural Inf. Process. Syst.*, 2023, pp. 58088–58112.
[70] H. Dong et al., "On the equivalence of decoupled graph convolution network and label propagation," in *Proc. Web Conf.*, 2021, pp. 3651–3662.
[71] Q. He, J. Chen, H. Xu, and K. He, "Structural robust label propagation on homogeneous graphs," in *Proc. IEEE Int. Conf. Data Mining*, 2022, pp. 181–190.
[72] J. Chen, B. Li, Q. He, and K. He, "PAMT: A novel propagation-based approach via adaptive similarity mask for node classification," *IEEE Trans. Computat. Social Syst.*, vol. 11, no. 5, pp. 5973–5983, Oct. 2024.
[73] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2847–2856.
[74] F. Feng, X. He, J. Tang, and T.-S. Chua, "Graph adversarial training: Dynamically regularizing based on graph structure," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 6, pp. 2493–2504, Jun. 2021.
[75] K. Sun, Z. Lin, H. Guo, and Z. Zhu, "Virtual adversarial training on graph convolutional networks in node classification," in *Proc. 2nd Chin. Conf. Pattern Recognit. Comput. Vis.*, 2019, pp. 431–443.
[76] Z. Deng, Y. Dong, and J. Zhu, "Batch virtual adversarial training for graph convolutional networks," *AI Open*, vol. 4, pp. 73–79, 2023.
[77] Y. Abbahaddou, S. Ennadir, J. F. Lutzeyer, M. Vazirgiannis, and H. Boström, "Bounding the expected robustness of graph neural networks subject to node feature attacks," in *Proc. 12th Int. Conf. Learn. Representations*, 2024.
[78] C. Huo, D. Jin, Y. Li, D. He, Y.-B. Yang, and L. Wu, "T2-GNN: Graph neural networks for graphs with incomplete features and structure via teacher-student distillation," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 4339–4346.
[79] S. Li, Y. Liu, Q. Chen, G. I. Webb, and S. Pan, "Noise-resilient unsupervised graph representation learning via multi-hop feature quality estimation," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2024, pp. 1255–1265.
[80] Y. Wang and Y. Yang, "Bayesian robust graph contrastive learning," 2022, *arXiv:2205.14109*.
[81] I. E. Olatunji, W. Nejdl, and M. Khosla, "Membership inference attack on graph neural networks," in *Proc. IEEE Int. Conf. Trust, Privacy Secur. Intell. Syst. Appl.*, 2021, pp. 11–20.
[82] X. He, R. Wen, Y. Wu, M. Backes, Y. Shen, and Y. Zhang, "Node-level membership inference attacks against graph neural networks," 2021, *arXiv:2102.05429*.
[83] V. Duddu, A. Boutet, and V. Shejwalkar, "Quantifying privacy leakage in graph embedding," in *Proc. Int. Conf. Mobile Ubiquitous Syst.: Comput., Netw. Serv.*, 2020, pp. 76–85.
[84] Z. Zhang et al., "GraphMI: Extracting private graph data from graph neural networks," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, 2021, pp. 3749–3755.
[85] B. Wu, X. Yang, S. Pan, and X. Yuan, "Model extraction attacks on graph neural networks: Taxonomy and realization," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2022, pp. 337–350.
[86] Y. Shen, X. He, Y. Han, and Y. Zhang, "Model stealing attacks against inductive graph neural networks," in *Proc. IEEE Symp. Secur. Privacy*, 2022, pp. 1175–1192.
[87] D. Xu, S. Yuan, X. Wu, and H. Phan, "DPNE: Differentially private network embedding," in *Proc. Adv. Knowl. Discov. Data Mining: 22nd Pacific-Asia Conf.*, 2018, pp. 235–246.
[88] I. E. Olatunji, T. Funke, and M. Khosla, "Releasing graph neural networks with differential privacy guarantees," *Trans. Mach. Learn. Res.*, 2023.
[89] T. T. Mueller, J. C. Paetzold, C. Prabhakar, D. Usynin, D. Rueckert, and G. Kaissis, "Differentially private graph neural networks for whole-graph classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 6, pp. 7308–7318, Jun. 2023.
[90] S. Sajadmanesh and D. Gatica-Perez, "Locally private graph neural networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 2130–2145.
[91] S. Zhang, H. Yin, T. Chen, Z. Huang, L. Cui, and X. Zhang, "Graph embedding for recommendation against attribute inference attacks," in *Proc. Web Conf.*, 2021, pp. 3002–3014.
[92] P. Liao et al., "Information obfuscation of graph neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 6600–6610.
[93] 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. 6904–6915, Apr. 2021.
[94] H. Hu, L. Cheng, J. P. Vap, and M. Borowczak, "Learning privacy-preserving graph convolutional network with partially observed sensitive attributes," in *Proc. ACM Web Conf.*, 2022, pp. 3552–3561.
[95] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, and S. Avestimehr, "Spreadgnn: Decentralized multi-task federated learning for graph neural networks on molecular data," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 6, 2022, pp. 6865–6873.
[96] Y. Pei et al., "Decentralized federated graph neural networks," in *Proc. Int. Workshop Federated Transfer Learn. Data Sparsity Confidentiality Conjunction IJCAI*, 2021.
[97] M. Chen, Z. Zhang, T. Wang, M. Backes, M. Humbert, and Y. Zhang, "Graph unlearning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2022, pp. 499–513.
[98] M. Stadler, B. Charpentier, S. Geisler, D. Zügner, and S. Günnemann, "Graph posterior network: Bayesian predictive uncertainty for node classification," in *Proc . Adv. Neural Inf. Process. Syst.*, 2021, pp. 18033–18048.
[99] Q. Wu, Y. Chen, C. Yang, and J. Yan, "Energy-based out-of-distribution detection for graph neural networks," in *Proc. Int. Conf. Learn. Representations*, 2023.
[100] Y. Song and D. Wang, "Learning on graphs with out-of-distribution nodes," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 1635–1645.
[101] T. Huang, D. Wang, and Y. Fang, "End-to-end open-set semi-supervised node classification with out-of-distribution detection," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 2087–2093.
[102] Y. Guo, C. Yang, Y. Chen, J. Liu, C. Shi, and J. Du, "A data-centric framework to endow graph neural networks with out-of-distribution detection ability," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2023, pp. 638–648.
[103] J. Tang, J. Li, Z. Gao, and J. Li, "Rethinking graph neural networks for anomaly detection," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 21076–21089.
[104] X. Zhao, F. Chen, S. Hu, and J.-H. Cho, "Uncertainty aware semi-supervised learning on graph data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 12827–12836.
[105] G. Zhang et al., "Dual-discriminative graph neural network for imbalanced graph-level anomaly detection," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 24144–24157.
[106] R. Ma, G. Pang, L. Chen, and A. van den Hengel, "Deep graph-level anomaly detection by glocal knowledge distillation," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2022, pp. 704–714.
[107] Y. Liu, K. Ding, H. Liu, and S. Pan, "GOOD-D: On unsupervised graph out-of-distribution detection," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2023, pp. 339–347.
[108] J. Duan et al., "Graph anomaly detection via multi-scale contrastive learning networks with augmented view," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 7459–7467.
[109] X. Luo et al., "Deep graph level anomaly detection with contrastive learning," *Sci. Rep.*, vol. 12, no. 1, 2022, Art. no. 19867.
[110] Z. Li, Q. Wu, F. Nie, and J. Yan, "GraphDE: A generative framework for debiased learning and out-of-distribution detection on graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 30277–30290.
[111] L. Zhao and L. Akoglu, "On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights," *Big Data*, vol. 11, no. 3, pp. 151–180, 2023.
[112] C. Qiu, M. Kloft, S. Mandt, and M. Rudolph, "Raising the bar in graph-level anomaly detection," in *Proc. Int. Joint Conf. Artif. Intell.*, 2023, pp. 2196–2203.
[113] L. Wang et al., "GOODAT: Towards test-time graph out-of-distribution detection," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 15537–15545.

[114] Y. Sui, X. Wang, J. Wu, M. Lin, X. He, and T.-S. Chua, "Causal attention for interpretable and generalizable graph classification," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 1696–1705.
[115] Y. Chen et al., "Learning causally invariant representations for out-of-distribution generalization on graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 22131–22148.
[116] S. Fan, X. Wang, C. Shi, P. Cui, and B. Wang, "Generalizing graph neural networks on out-of-distribution graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 1, pp. 322–337, Jan. 2024.
[117] D. Buffelli, P. Liò, and F. Vandin, "SizeShiftReg: A regularization method for improving size-generalization in graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 31871–31885.
[118] H. Li, Z. Zhang, X. Wang, and W. Zhu, "Learning invariant graph representations for out-of-distribution generalization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 11828–11841.
[119] N. Yang, K. Zeng, Q. Wu, X. Jia, and J. Yan, "Learning substructure invariance for out-of-distribution molecular representations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 12964–12978.
[120] J. Yu, J. Liang, and R. He, "Mind the label shift of augmentation-based graph OOD generalization," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 11620–11630.
[121] Q. Wu, H. Zhang, J. Yan, and D. Wipf, "Handling distribution shifts on graphs: An invariance perspective," in *Proc. Int. Conf. Learn. Representations*, 2022.
[122] H. Xue et al., "CAP: Co-adversarial perturbation on weights and features for improving generalization of graph neural networks," 2021, *arXiv:2110.14855*.
[123] Y. Sui et al., "Unleashing the power of graph data augmentation on covariate distribution shift," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 18109–18131.
[124] S. Gui, M. Liu, X. Li, Y. Luo, and S. Ji, "Joint learning of label and environment causal independence for graph out-of-distribution generalization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 3945–3978.
[125] Y. Wu, A. Bojchevski, and H. Huang, "Adversarial weight perturbation improves generalization in graph neural networks," in *Proc. Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 10417–10425.
[126] N. Yin et al., "Deal: An unsupervised domain adaptive framework for graph-level classification," in *Proc. ACM Int. Conf. Multimedia*, 2022, pp. 3470–3479.
[127] Y. Zhang et al., "A survey on privacy in graph neural networks: Attacks, preservation, and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 12, pp. 7497–7515, Dec. 2024.
[128] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[129] H. Li, X. Wang, Z. Zhang, and W. Zhu, "OOD-GNN: Out-of-distribution generalized graph neural network," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 7, pp. 7328–7340, Jul. 2022.
[130] M. Wu, S. Pan, X. Zhu, C. Zhou, and L. Pan, "Domain-adversarial graph neural networks for text classification," in *Proc. IEEE Int. Conf. Data Mining*, 2019, pp. 648–657.
[131] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 3734–3743.
[132] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 4805–4815.
[133] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 4438–4445.
[134] W. Ju et al., "Zero-shot node classification with graph contrastive embedding network," *Trans. Mach. Learn. Res.*, 2023.
[135] S. Agarwal, "Ranking on graph data," in *Proc. Int. Conf. Mach. Learn.*, 2006, pp. 25–32.
[136] S. Yi et al., "Redundancy-free self-supervised relational learning for graph clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 12, pp. 18313–18327, Dec. 2024.
[137] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5171–5181.
[138] W. Ju et al., "TGNN: A joint semi-supervised framework for graph-level classification," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 2122–2128.
[139] W. Ju et al., "Hypergraph-enhanced dual semi-supervised graph classification," in *Proc. Int. Conf. Mach. Learn.*, 2024, pp. 22594–22604.
[140] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola, "Learning graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1048–1058, Jun. 2009.
[141] X. Guo and L. Zhao, "A systematic survey on deep generative models for graph generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5370–5390, May 2023.
[142] Y. Liu et al., "Pick and choose: A GNN-based imbalanced learning approach for fraud detection," in *Proc. Web Conf.*, 2021, pp. 3168–3177.
[143] Y. Liu, Z. Gao, X. Liu, P. Luo, Y. Yang, and H. Xiong, "QTIAH-GNN: Quantity and topology imbalance-aware heterogeneous graph neural network for bankruptcy prediction," in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2023, pp. 1572–1582.
[144] C. Zhang et al., "When sparsity meets contrastive models: Less graph data can bring better class-balanced representations," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 41133–41150.
[145] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9268–9277.
[146] Q. Sun et al., "Position-aware structure learning for graph topology-imbalance by relieving under-reaching and over-squashing," in *Proc. ACM Int. Conf. Inf. Knowl. Manage.*, 2022, pp. 1848–1857.
[147] X. Luo et al., "CLEAR: Cluster-enhanced contrast for self-supervised graph representation learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 899–912, Jan. 2024.
[148] W. Ju et al., "Unsupervised graph-level representation learning with hierarchical contrasts," *Neural Netw.*, vol. 158, pp. 359–368, 2023.
[149] Y. Jin and X. Zhu, "An systematic study and analysis of graph neural networks under noise," *ACM Trans. Knowl. Discov. Data*, vol. 19, 2025, Art. no. 113.
[150] H. NT, C. J. Jin, and T. Murata, "Learning graph neural networks with noisy labels," 2019, *arXiv:1905.01591*.
[151] N. Yin, L. Shen, M. Wang, X. Luo, Z. Luo, and D. Tao, "OMG: Towards effective graph classification against label noise," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 12, pp. 12873–12886, Dec. 2023.
[152] N. Yin, L. Shen, C. Chen, X.-S. Hua, and X. Luo, "Sport: A subgraph perspective on graph classification with label noise," *ACM Trans. Knowl. Discov. Data*, vol. 18, no. 9, pp. 1–20, 2024.
[153] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.
[154] X. Luo et al., "Self-supervised graph-level representation learning with adversarial contrastive learning," *ACM Trans. Knowl. Discov. Data*, vol. 18, 2023, Art. no. 34.
[155] W. Ju et al., "GPS: Graph contrastive learning via multi-scale augmented views from adversarial pooling," *Sci. China Inf. Sci.*, vol. 68, no. 1, 2025, Art. no. 112101.
[156] W. Ju et al., "Towards graph contrastive learning: A survey and beyond," 2024, *arXiv:2405.11868*.
[157] X. Li et al., "Contrastive learning of graphs under label noise," *Neural Netw.*, vol. 172, 2024, Art. no. 106113.
[158] X. Yu, B. Han, J. Yao, G. Niu, I. Tsang, and M. Sugiyama, "How does disagreement help generalization against label corruption?," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7164–7173.
[159] Y. Ma, H. Derksen, W. Hong, and J. Wright, "Segmentation of multivariate mixed data via lossy data coding and compression," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 9, pp. 1546–1562, Sep. 2007.
[160] K. Xu et al., "Topology attack and defense for graph neural networks: An optimization perspective," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, 2019, pp. 3961–3967.
[161] J. Fox and S. Rajamanickam, "How robust are graph neural networks to structural noise?," 2019, *arXiv:1912.10206*.
[162] C. A. R. de Sousa, S. O. Rezende, and G. E. Batista, "Influence of graph construction on semi-supervised learning," in *Proc. Joint Eur. Conf. Conf. Mach. Learn. Knowl. Discov. Databases*, 2013, pp. 160–175.
[163] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.
[164] L. Sun et al., "Adversarial attack and defense on graph data: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 7693–7711, Aug. 2023.
[165] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," *IEEE Trans. Signal Process.*, vol. 68, pp. 5680–5695, 2020.
[166] G. Wan and H. Kokel, "Graph sparsification via meta-learning," in *Proc. DLG, Assoc. Advance. Artif. Intell.*, 2021.

[167] H. Dai et al., “Adversarial attack on graph structured data,” in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1115–1124.
[168] W. Ju et al., “GLCC: A general framework for graph-level clustering,” in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 4391–4399.
[169] Y. Zhao, X. Luo, W. Ju, C. Chen, X.-S. Hua, and M. Zhang, “Dynamic hypergraph structure learning for traffic flow forecasting,” in *Proc. Int. Conf. Data Eng.*, 2023, pp. 2303–2316.
[170] M. Jagielski, J. Ullman, and A. Oprea, “Auditing differentially private machine learning: How private is private SGD?,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 22205–22216.
[171] X. He, J. Jia, M. Backes, N. Z. Gong, and Y. Zhang, “Stealing links from graph neural networks,” in *Proc. USENIX Secur. Symp.*, 2021, pp. 2669–2686.
[172] B. Wu, X. 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*, 2021, pp. 1421–1426.
[173] D. DeFazio and A. Ramesh, “Adversarial model extraction on graph neural networks,” 2019, *arXiv:1912.07721*.
[174] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, “Disentangled graph collaborative filtering,” in *Proc. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 1001–1010.
[175] J. Xu, S. Koffas, O. Ersoy, and S. Picek, “Watermarking graph neural networks based on backdoor attacks,” in *Proc. Eur. Symp. Secur. Privacy*, 2023, pp. 1179–1197.
[176] X. Zhao, H. Wu, and X. Zhang, “Watermarking graph neural networks by random graphs,” in *Proc. Int. Symp. Digit. Forensics Secur.*, 2021, pp. 1–6.
[177] Z. Li and Y. Zhang, “Membership leakage in label-only exposures,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2021, pp. 880–895.
[178] Z. Zhang, M. Chen, M. Backes, Y. Shen, and Y. Zhang, “Inference attacks against graph neural networks,” in *Proc. USENIX Secur. Symp.*, 2022, pp. 4543–4560.
[179] Z. Zhang, Q. Liu, Z. Huang, H. Wang, C.-K. Lee, and E. Chen, “Model inversion attacks against graph neural networks,” *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 9, pp. 8729–8741, Sep. 2023.
[180] F. Wu, Y. Long, C. Zhang, and B. Li, “LinkTeller: Recovering private edges from graph neural networks via influence analysis,” in *Proc. IEEE Symp. Secur. Privacy*, 2022, pp. 2005–2024.
[181] I. E. Olatunji, M. Rathee, T. Funke, and M. Khosla, “Private graph extraction via feature explanations,” in *Proc. Privacy Enhancing Technol.*, vol. 2023, no. 2, 2023.
[182] X. Niu et al., “A dual heterogeneous graph attention network to improve long-tail performance for shop search in E-commerce,” in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 3405–3415.
[183] Y. Rong et al., “Self-supervised graph transformer on large-scale molecular data,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 12559–12571.
[184] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Proc. 3rd Theory Cryptogr. Conf.*, 2006, pp. 265–284.
[185] I.-C. Hsieh and C.-T. Li, “NetFense: Adversarial defenses against privacy attacks on neural networks for graph data,” *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 796–809, Jan. 2023.
[186] P. Kairouz et al., “Advances and open problems in federated learning,” *Found. Trends Mach. Learn.*, vol. 14, no. 1/2, pp. 1–210, 2021.
[187] C. He et al., “FedGraphNN: A federated learning system and benchmark for graph neural networks,” 2021, *arXiv:2104.07145*.
[188] Y. Yao, W. Jin, S. Ravi, and C. Joe-Wong, “FedGCN: Convergence-communication tradeoffs in federated training of graph convolutional networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 79748–79760.
[189] H. Zhang, T. Shen, F. Wu, M. Yin, H. Yang, and C. Wu, “Federated graph learning–A position paper,” 2021, *arXiv:2105.11099*.
[190] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” in *Proc. Int. Conf. Learn. Representations*, 2017.
[191] Y. Liu, K. Ding, Q. Lu, F. Li, L. Y. Zhang, and S. Pan, “Towards self-interpretable graph-level anomaly detection,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 8975–8987.
[192] Z. Ding et al., “SGOOD: Substructure-enhanced graph-level out-of-distribution detection,” in *Proc. 33rd ACM Int. Conf. Inf. Knowl. Manage.*, 2024, pp. 467–476.
[193] Y. Wu, X. Wang, A. Zhang, X. He, and T.-S. Chua, “Discovering invariant rationales for graph neural networks,” in *Proc. Int. Conf. Learn. Representations*, 2022.
[194] Y. Liu et al., “Flood: A flexible invariant learning framework for out-of-distribution generalization on graphs,” in *Proc. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2023, pp. 1548–1558.
[195] Y. Qin, X. Wang, Z. Zhang, P. Xie, and W. Zhu, “Graph neural architecture search under distribution shifts,” in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 18083–18095.
[196] Y. Zhou, G. Kutyniok, and B. Ribeiro, “OOD link prediction generalization capabilities of message-passing GNNs in larger test graphs,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 20257–20272.
[197] W. Ju, J. Yang, M. Qu, W. Song, J. Shen, and M. Zhang, “KGNN: Harnessing kernel-based networks for semi-supervised graph classification,” in *Proc. ACM Int. Conf. Web Search Data Mining*, 2022, pp. 421–429.
[198] S. Gui, X. Li, L. Wang, and S. Ji, “Good: A graph out-of-distribution benchmark,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 2059–2073.
[199] H. Li, X. Wang, Z. Zhang, and W. Zhu, “Out-of-distribution generalization on graphs: A survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2025.
[200] J.-B. Grill et al., “Bootstrap your own latent-A new approach to self-supervised learning,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 21271–21284, 2020.
[201] M. Yi et al., “Improved OOD generalization via adversarial training and pretraing,” in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 11987–11997.
[202] M. Wu, S. Pan, C. Zhou, X. Chang, and X. Zhu, “Unsupervised domain adaptive graph convolutional networks,” in *Proc. Web Conf.*, 2020, pp. 1457–1467.
[203] X. Luo et al., “RIGNN: A rationale perspective for semi-supervised open-world graph classification,” *Trans. Mach. Learn. Res.*, 2023.
[204] X. Luo, Y. Zhao, Y. Qin, W. Ju, and M. Zhang, “Towards semi-supervised universal graph classification,” *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 1, pp. 416–428, Jan. 2024.
[205] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong, “Backdoor attacks to graph neural networks,” in *Proc. ACM Symp. Access Control Models Technol.*, 2021, pp. 15–26.
[206] Z. Chen, S. Villar, L. Chen, and J. Bruna, “On the equivalence between graph isomorphism testing and function approximation with GNNs,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 15894–15902.
[207] Anonymous, “GraphGuard: Provably robust graph classification against adversarial attacks,” in *Proc. Int. Conf. Learn. Representations*, 2024.
[208] M. Réau, N. Renaud, L. C. Xue, and A. M. Bonvin, “Deeprank-GNN: A graph neural network framework to learn patterns in protein–protein interfaces,” *Bioinformatics*, vol. 39, no. 1, 2023, Art. no. btac759.
[209] Z. Yu, Y. Guo, and Y. Chen, “Learning trajectory routing with graph neural networks,” in *Proc. Int. Conf. Big Data Comput.*, 2020, pp. 121–126.

Wei Ju (Member, IEEE) received the PhD degree with the School of Computer Science, Peking University, China, in 2022. He was a postdoc with Peking University. He is currently an associate professor with the College of Computer Science, Sichuan University. He has authored or coauthored more than 80 papers in top-tier venues. His research interests include graph neural networks, bioinformatics, drug discovery, recommender systems, and spatio-temporal analysis.

Siyu Yi (Member, IEEE) received the PhD degree in statistics from Nankai University, in 2024. She is currently a postdoc in mathematics with Sichuan University. She has authored or coauthored more than 20 papers. Her research interests include graph machine learning, statistical learning, and subsampling in Big Data.

Yifan Wang received the PhD degree in computer science from Peking University, in 2023. He is currently an assistant professor with the School of Information Technology $\&amp;$ Management, University of International Business and Economics. His research interests include graph neural networks, disentangled representation learning, drug discovery, and recommender systems.

Zhiping Xiao received the PhD degree from Computer Science Department, University of California at Los Angeles, Los Angeles, CA, USA. She is currently a postdoc with the Paul G. Allen School of Computer Science and Engineering, University of Washington. Her research interests include graph representation learning and social network analysis.

Zhengyang Mao is currently working toward the master's degree with the School of Computer Science, Peking University. His research interests include graph representation learning, long-tailed learning, and quantitative finance.

Hourun Li is currently a master's student at the School of Computer Science, Peking University. His research interests include graph representation learning, out-of-distribution detection and recommender systems.

Yiyang Gu received the BS degree in computer science from Peking University, Beijing, China, in 2021. He is currently working toward the PhD degree in computer science with Peking University. His research interests include data mining, graph machine learning, self-supervised learning, and bioinformatics.

Yifang Qin received the BS degree from School of EECS, Peking University, Beijing, China, where he is currently working toward the Graduate degree with the School of Computer Science, Peking University. His research interests include graph representation learning and recommender systems.

Nan Yin received the PhD degree from the School of Computer Science and Technology, National University of Defense Technology. He is currently a postdoc with the Mohamed bin Zayed University of Artificial Intelligence. His research interests includes transfer learning and graphs.

Senzhang Wang (Member, IEEE) received the PhD degree from Beihang University, China. He is currently a professor with the School of Computer Science and Engineering, Central South University, Changsha. He has authored or coauthored more than 100 papers on the top international journals and conferences. His research interests include data mining and social network analysis.

Xinwang Liu (Senior Member, IEEE) received the PhD degree from the National University of Defense Technology (NUDT), China, in 2013. He is currently a professor with the School of Computer, NUDT. He has authored or coauthored more than 200 peer-reviewed papers, including those in highly regarded journals and conferences such as *IEEE Transactions on Pattern Analysis and Machine Intelligence*., *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Image Processing*, *IEEE Transactions on Neural Networks and Learning Systems* (IEEE TNNLS), ICML, NeurIPS, CVPR, AAAI, and IJCAI. His research interests include kernel learning, multi-view clustering, and unsupervised feature learning. He is also an associate editor for IEEE T-NNLS and *Information Fusion Journal*.

Philip S. Yu (Life Fellow, IEEE) received the PhD degree in electrical engineering from Stanford University, USA. He is currently a distinguished professor of computer science with the University of Illinois at Chicago. He was the recipient of ACM SIGKDD 2016 Innovation Award for his influential research and scientific contributions on mining, fusion and anonymization of Big Data, IEEE Computer Societys 2013 Technical Achievement Award for pioneering and fundamentally innovative contributions to the scalable indexing, querying, searching, mining and anonymization of Big Data. From 2011 to 2017, he was the editor-in-chiefs of *ACM Transactions on Knowledge Discovery from Data* and *IEEE Transactions on Knowledge and Data Engineering* from 2001 to 2004.

Ming Zhang (Member, IEEE) received the PhD degree in computer science from Peking University. She is currently a full professor with the School of Computer Science, Peking University. She has authored or coauthored more than 200 research papers on Text Mining and Machine Learning in the top journals and conferences. Prof. Zhang was the recipient of the the Best Paper of ICML 2014 and Best Paper nominee of WWW 2016 and ICDM 2022. She is also a member with the Advisory Committee of Ministry of Education, China and the chair with ACM SIGCSE China. She is one of the fifteen members of ACM/IEEE CC2020 Steering Committee.