Federated Graph Neural Networks: Overview, Techniques, and Challenges

Rui Liu, Pengwei Xing, Zichao Deng, Anran Li, Cuntai Guan, *Fellow, IEEE*, and Han Yu, *Senior Member, IEEE*

***Abstract*— Graph neural networks (GNNs) have attracted extensive research attention in recent years due to their capability to progress with graph data and have been widely used in practical applications. As societies become increasingly concerned with the need for data privacy protection, GNNs face the need to adapt to this new normal. Besides, as clients in federated learning (FL) may have relationships, more powerful tools are required to utilize such implicit information to boost performance. This has led to the rapid development of the emerging research field of federated GNNs (FedGNNs). This promising interdisciplinary field is highly challenging for interested researchers to grasp. The lack of an insightful survey on this topic further exacerbates the entry difficulty. In this article, we bridge this gap by offering a comprehensive survey of this emerging field. We propose a 2-D taxonomy of the FedGNN literature: 1) the main taxonomy provides a clear perspective on the integration of GNNs and FL by analyzing how GNNs enhance FL training as well as how FL assists GNN training and 2) the auxiliary taxonomy provides a view on how FedGNNs deal with heterogeneity across FL clients. Through discussions of key ideas, challenges, and limitations of existing works, we envision future research directions that can help build more robust, explainable, efficient, fair, inductive, and comprehensive FedGNNs.**

***Index Terms*— Federated learning (FL), graph neural networks (GNNs).**

I. Introduction

GRAPH neural networks (GNNs) are powerful tools for dealing with graph-structured data [1]. Graph-structured data are data samples connected by a graph topology. For example, molecular data are graph-structured data in which atoms serve as nodes and the bonds connecting them serve as edges in the graph. GNNs can improve the quality of node embedding by considering neighborhood information extracted from the underlying graph topology. They have been widely adopted by diverse applications including drug discovery [2], neuroscience [3], social networks [4], knowledge graphs (KGs) [5], recommender systems [6], and traffic flow prediction [7].

A well-trained GNN model requires a large amount of training graph data, which may be distributed among multiple data owners in practice. Due to privacy concerns [8], these data owners (also known as clients) may not be willing to share their private data, which leads to the problem of data isolation. Furthermore, the graph data stored by different clients are often non-independent identically distributed (non-i.i.d.), which exacerbates the data isolation issue. This non-i.i.d. property can manifest itself as differences in graph structures or node feature distributions across clients.

Federated learning (FL), a distributed collaborative machine learning (ML) paradigm, is a promising approach to deal with the data isolation issue [9], [10], [11], [12], [13]. It enables local models to benefit from each other while keeping local data private [14], [15]. Besides, the problem of learning personalized FL models in the presence of non-i.i.d. data has been extensively studied [16]. In FL, only model parameters or embedding features are shared between participants without exposing potentially sensitive local data. This architectural design, combined with various cryptographic techniques, can provide effective protection of local data privacy. In some situations, there are relationships between FL participants, consisting of a graph topology with participants as nodes. This relationship graph may contain useful but implicit information, such as the participants' similarities and trust. Utilizing the topology information to boost FL performance remains a challenge.

The confluence of these trends of development has inspired the emergence of the field of federated GNNs (FedGNNs) [17], which has witnessed rapid development in recent years. Existing works such as [18] and [19] summarize FedGNNs into three categories: 1) FL clients containing multiple graphs; 2) FL clients containing subgraphs; and 3) FL clients containing one node according to the distribution of graph data. However, as technical research for these envisioned categories had not been extensively studied at the time, these early positioning papers only provided general ideas without specific works or problem descriptions for the different categories. Besides, there can be overlapping situations between Category 2 and Category 3. For example, in Category 2, if there are some edges connecting subgraphs residing in different FL clients, these edges can be regarded as interclient graphs, which also exist in Category 3. A recent survey [20] simplifies the three-category taxonomy with a two-category taxonomy based

Manuscript received 21 December 2022; revised 19 May 2023 and 22 November 2023; accepted 23 January 2024. This work was supported in part by the National Research Foundation, Singapore, and Defence Science Organisation (DSO) National Laboratories through the AI Singapore Programme under AISG Award AISG2-RP-2020-019; in part by the Alibaba Group through the Alibaba Innovative Research (AIR) Program and the Alibaba-NTU Singapore Joint Research Institute (JRI) (Alibaba-NTU-AIR2019B1), Nanyang Technological University (NTU), Singapore; in part by the RIE 2020 Advanced Manufacturing and Engineering (AME) Programmatic Fund, Singapore, under Grant A20G8b0102; in part by the Nanyang Technological University through the Nanyang Assistant Professorship (NAP); and in part by the Future Communications Research and Development Programme under Grant FCP-NTU-RG-2021-014. *(Corresponding author: Han Yu.)*

The authors are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: rui.liu@ntu.edu.sg; han.yu@ntu.edu.sg).

Digital Object Identifier 10.1109/TNNLS.2024.3360429

on the location of structural information: 1) structural information existing in the FL clients and 2) structural information existing between FL clients. However, the problem-based subcategories of the taxonomy are inadequate as they only cover a limited subset of the FedGNN literature. Furthermore, this work primarily focuses on summarizing the existing literature, without delving into an insightful analysis of the advantages and disadvantages of the current approaches.

Currently, there is a lack of a comprehensive survey on FedGNNs that provide an insightful view on this critical topic for new researchers. This article bridges this important gap. The main contributions are given as follows.

1) We propose a 2-D taxonomy that categorizes existing works on FedGNNs from two perspectives.
   a) *Main Taxonomy:* How FL and GNNs are integrated together.
   b) *Auxiliary Taxonomy:* How FedGNNs deal with heterogeneity across FL clients. We highlight the challenges, specific methods, and potential limitations for each category.
2) We discuss commonly adopted public datasets and evaluation metrics in the existing literature for FedGNN benchmarking and provide suggestions on enhancing FedGNN experiment design.
3) We envision promising future directions of research toward building more robust, explainable, efficient, fair, inductive, and comprehensive FedGNNs to enhance the trustworthiness of this field.

The rest of this article is organized as follows. Key terminologies used in FedGNNs and the proposed 2-D taxonomy are introduced in Section II. The main challenges, techniques, and limitations of FedGNNs are reviewed in Sections III and IV for the main taxonomy and in Section V for the auxiliary taxonomy. Section VI summarizes applications, datasets, evaluation metrics, and data partition methods of FedGNNs. Finally, we propose seven future directions toward building trustworthy FedGNNs in Section VII. Section VIII concludes this survey.

II. Terminology and Taxonomy

This section explains key terminologies in GNNs and FL and introduces the proposed 2-D FedGNN taxonomy.

A. Terminology

GNNs are a class of deep learning models designed to perform feature embedding and inference on graph data. They require two inputs: 1) a graph, which consists of *nodes* and *edges*, represented by an *adjacency matrix* $\mathbf{A} \in \mathbb{R}^{N\times N}$, and 2) their *node features* $\mathbf{X} \in \mathbb{R}^{N\times f}$, where N denotes the number of nodes and f is the number of node features. GNNs update the embedding of a given node by aggregating information from its neighboring nodes with the following function:

$$\mathbf{X}^{(l+1)} = \mathrm{GNN}\left(\mathbf{A}, \mathbf{X}^{(l)}, \mathbf{w}^{(l)}\right) \tag{1}$$

where $\mathrm{GNN}(\cdot)$ indicates the graph aggregation function, which can be the mean, weighted average, or max/min pooling methods. $\mathbf{X}^{(l)}$ and $\mathbf{X}^{(l+1)}$ represent the node embedding in the lth and $(l+1)$th layers. $\mathbf{A}$ denotes the graph adjacency matrix. $\mathbf{w}^{(l)}$ are the trainable model weights in the lth layer.

FL is a collaborative ML paradigm that trains a model across multiple data owners, without exchanging raw data. It has two main settings: *horizontal* FL (HFL) and *vertical* FL (VFL) [21]. In HFL, the datasets in different clients have large overlaps in the feature space but little overlap in the sample space. In VFL, the clients have little overlap in the feature space but large overlaps in the sample space. In FL, data owners with sensitive local data can be referred to as *clients* if they are coordinated by a central entity referred to as the *server*. Under the HFL setting, based on the communication architecture, FL has two settings: *centralized* FL and *decentralized* FL. In centralized FL, the server coordinates the clients to jointly learn a model, while in decentralized FL, clients communicate with each other without a centralized server to jointly learn a model.

FL also involves an "aggregation" operation. Aggregation, in the context of FL, updates model parameters in the server with local model parameters uploaded by clients. For instance, it can be achieved with an averaging operation following FedAvg [22]:

$$\mathbf{w}^{r} = \sum_{c=1}^{C} \frac{n_c}{n} \mathbf{w}_c^{r} \tag{2}$$

where n_c denotes the number of samples in client c and n represents the total number of samples in all clients. $\mathbf{w}^r$ and $\mathbf{w}_c^r$ are the global model weights and the local model weights in the cth client at round r.

To disambiguate between the aggregation operations in GNNs and FL, we refer to them as *GNN aggregation* and *FL aggregation*, respectively, in this article.

B. Proposed 2-D FedGNN Taxonomy

As shown in Fig. 1, the proposed FedGNN taxonomy consists of two dimensions. The first dimension focuses on the integration of FL and GNNs, which serves as the main taxonomy. The second dimension focuses on FL aggregation solutions dealing with different levels of graph data heterogeneity, which serves as the auxiliary taxonomy.

The main taxonomy focuses on dealing with the integration of FL and GNNs. According to the existence of explicit graph information, the main taxonomy can be divided into two main categories: *GNN-assisted FL* and *FL-assisted GNNs*. These categories are further subdivided into various scenarios based on their layout settings. Each subscenario introduces general integration strategies along with detailed solutions for key challenges. To provide a succinct overview, we briefly introduce the two main categories of the main taxonomy.

1) GNN-Assisted FL: Methods under this category mainly target situations where there is no explicit graph, but implicit graph information exists. The implicit graph information can be applied to GNN models to assist FL model training. The implicit graph can be the connectivity network between clients, denoted by the black dotted lines in the *GNN-assisted FL* category of Fig. 1. FL model training is the main task, and the local data of clients are not necessarily graph data. Works in this category are further divided into two scenarios according

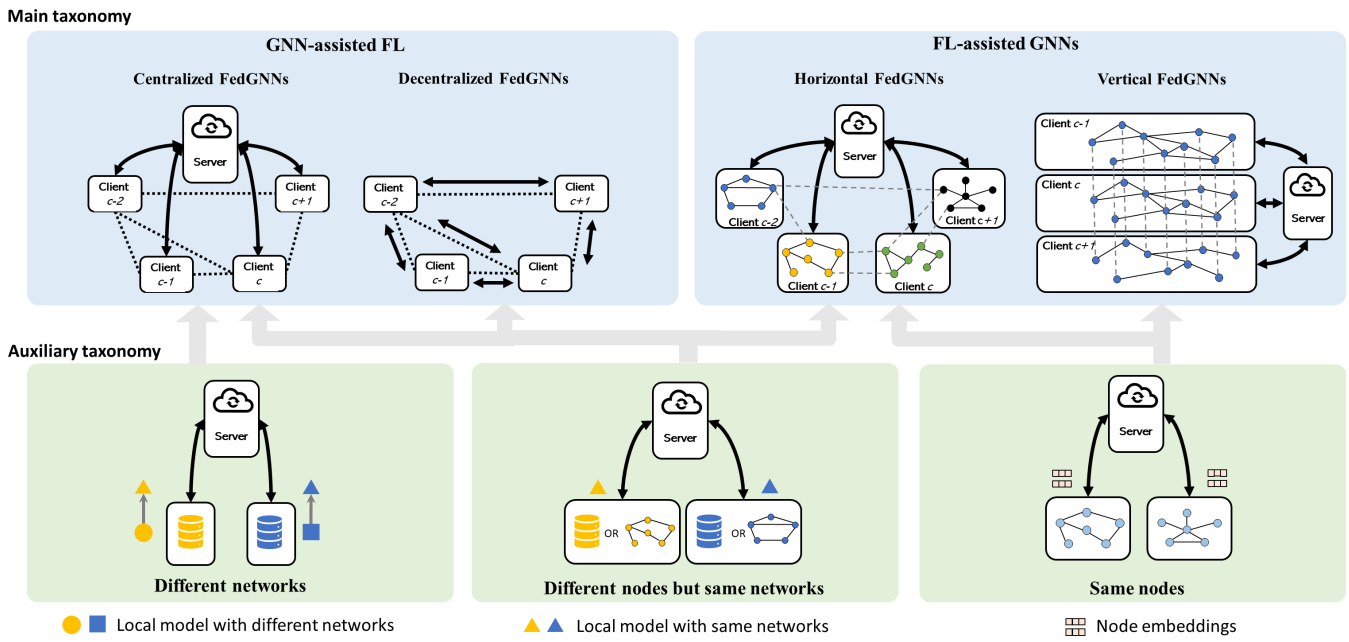

Fig. 1. Proposed 2-D taxonomy for FedGNN research.

to whether a central FL server exists or not: 1) *centralized FedGNNs* and 2) *decentralized FedGNNs*. *Centralized FedGNNs* have a central server to coordinate clients. Clients in *decentralized FedGNNs* communicate with their neighbors directly. Communications between entities are denoted by the black double arrows in Fig. 1.

2) FL-Assisted GNNs: Approaches under this category focus on the situation where explicit graph data exist and are isolated across clients, in which GNNs require FL algorithms to assist model training with such data. The explicit isolated graph data are denoted by the small graphs inside clients in the *FL-assisted GNN* category of Fig. 1. Training the GNN model with isolated graph data silos is the main task. Methods under this category are further divided into two scenarios according to the overlapping scale of graph node IDs across clients: 1) *horizontal FedGNNs* and 2) *vertical FedGNNs*. Clients in *horizontal FedGNNs* have graph data consisting of nodes that are largely not overlapping. Clients in *vertical FedGNNs* have graph data sharing the same set of node IDs. Nodes' overlapping is denoted by the gray dotted lines in Fig. 1.

The auxiliary taxonomy focuses on dealing with the heterogeneity among FL clients. They can be divided into three categories: 1) clients having nodes with the same IDs; 2) clients having different nodes but the same network structure; and 3) clients employing different network structures. Different intermediate information is applied in FL aggregation for different categories. For clients having the same nodes, node embedding features are uploaded to the FL server for aggregation. This can be found in vertical FedGNNs and some horizontal FedGNN works with overlapping nodes, indicated by the thick gray arrows in Fig. 1. For clients having different nodes but applying the same network structures for training, model weights and gradients are used for FL aggregation. This can be found in both scenarios of GNN-assisted FL and some horizontal FedGNN works without overlapping nodes. For the clients training local models with different network structures, the network structure can be modeled as a graph first, and a GNN model is applied to it. Then, the GNN model weights or gradients can be used for FL aggregation. Currently, this can only be found in centralized FedGNN works.

III. GNN-Assisted FL

In this section, we review approaches under the *GNN-assisted FL* category in the main taxonomy, dealing with the FL systems with implicit graph information. Such implicit graph information comes from two sources: 1) interclient graph and 2) layer connectivity in deep neural networks. For example, road traffic monitoring sensors that are near each other tend to record similar traffic conditions. An interclient graph can be built from such graph-structured clients with each of them represented as a node in the graph, indicated by the black dotted lines in Fig. 2. Due to the existence of the interclient graph, GNN algorithms are applied to assist the FL training process. Besides, GNNs can also assist the FL system by modeling the neural network as a graph when local model architectures are different across clients.

According to the existence of a central server in the FL system, *GNN-assisted FL* has two scenarios: *centralized FedGNNs* and *decentralized FedGNNs*. Since the *centralized FedGNNs* have both the server and clients, according to the location of the implicit graph, two subscenarios are generated: 1) implicit graph located in the server and 2) implicit graph located in the clients, referring to the left part in Fig. 2. Without a central server, the implicit graph in *decentralized FedGNNs* can be only distributed across the clients, referring to the right part in Fig. 2. For each subscenario, we first introduce the setting assumption with general solutions and then list the main challenges in bold titles followed by detailed techniques. Their advantages and disadvantages are summarized at the end of this section.

A. Centralized FedGNNs

Centralized FedGNNs have a central server to coordinate clients. Depending on the location of the implicit graph, GNN

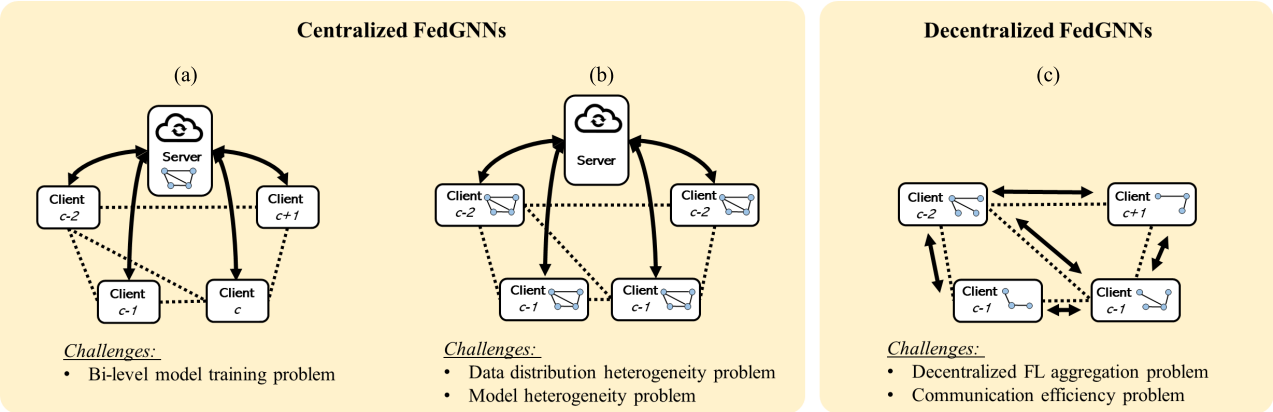

Fig. 2. Illustrations of subscenarios with corresponding challenges in the GNN-assisted FL category. (a) Implicit graph located in the server. (b) Implicit graph located in the clients. (c) Implicit graph distributed in the clients.

training with the graph can be performed at the server [see Fig. 2(a)] or local client [see Fig. 2(b)].

1) Implicit Graph Located in the Server: A GNN model is trained in the server with the interclient graph. It assumes that neighboring clients tend to have similar local models or feature embedding. The server first collects parameters from clients as it does in standard FL. The uploaded local model parameters are treated as the node features in the interclient graph. Then, it trains a GNN model with the uploaded parameters to facilitate FL aggregation. Finally, the updated parameters are sent back to the clients. The interclient graph can be given in advance or extracted with a self-attention module during training [23]. As the server has a separate GNN model, how to train both local models and the GNN model (bilevel model) in the server simultaneously is a challenge.

a) Bilevel model training problem: Works such as [24] and [25] design bilevel optimization schemes to train both local models and the GNN model with two types of objective functions: the local task objective functions $g_c(\cdot)$ for the local model training in the clients and an objective function $f(\cdot)$ for GNN model training in the FL server

$$\begin{aligned} \min_{\phi} \quad & f\left(\phi, \mathbf{w}_c^*(\phi) | c \in 1, \ldots, C\right) \\ \text{s.t.} \quad & \mathbf{w}_c^*(\phi) \in \arg\min_{\mathbf{w}_c} g_c(\phi, \mathbf{w}_c) \end{aligned} \tag{3}$$

where ϕ denotes the trainable parameters in the GNN model and $\mathbf{w}_c^*(\phi)$ is the local solution weight vector of client c. Big-Fed [24] and SFL [25] adopt different objective functions for $f(\cdot)$ to fulfill the assumption that neighboring clients' local models are similar. Big-Fed proposed an unsupervised contrastive learning loss function; meanwhile, SFL [25] proposed a supervised loss function with a graph smoothness regularization to train both local and global models.

Different from the bilevel optimization, some works [26], [27] train the local models and the GNN model sequentially with separate objective functions. For example, clients in [27] train their local models with different local tasks. Then, the server trains a GNN model to fuse multitask local estimates. By minimizing data reconstruction error with a graph regularization term, local estimates can be refined based on the clients' similarities. HFME [28] updates the local models with local loss functions and the global model with global loss functions iteratively. The GNN-based grouping predictor in the server is designed to cluster clients into groups based on their data distribution similarities. Clients in the same group train collaboratively to alleviate heterogeneity problems. PDGNet [26] models the power allocation policy with a GNN model in the server to find the optimal power allocation policy. The objective function is to minimize the transmission error probability for all FL clients. The GNN model is trained with a primal-dual iterative approach. CNNFGNN [29] and MLFGL [23] train the FL local models and the GNN model with only a local objective function. They perform alternating optimization to update clients' model weights with GNN model weights fixed and then update GNN model weights with the FL local model weights fixed, over multiple rounds.

2) Implicit Graph Located in the Clients: Depending on the implicit graph sources, GNN models are trained in the clients to solve two challenges: 1) data distribution heterogeneity and 2) model heterogeneity. The first challenge assumes that the implicit graph in the clients is the interclient graph indicating the relationships among clients. The second challenge assumes that the implicit graph in the clients is layer connectivity in the neural network. The paradigm follows the general FL training procedure where a GNN model is trained collaboratively by clients with model weights uploaded to the FL server. The server performs FL aggregation and distributes updated model weights to the clients for the next round of training. Meanwhile, building different graphs in the FL clients can solve different problems.

a) Data distribution heterogeneity problem: Under this setting, clients not only train the local models as they do in standard FL but also train a GNN model with the global graph to obtain global knowledge from other clients to address the data distribution heterogeneity issue. FedCG [30] builds a fully connected graph based on the similarity between clients' model weights or pattern features. A client trains a GNN with the graph to obtain the global embedding and then combines the local model embedding and the global embedding with a trainable weight.

b) Model heterogeneity problem: Standard FL is not well suited to deal with situations in which clients' local models are heterogeneous. HAFL-GHN [31] proposes a solution with the help of GNNs. It models the neural architecture in the client as a graph with each parametric layer as a vertex and the computational flows between layers as edges. The node

features are initialized with a categorical (one-hot) feature indicating the layer type. A GNN-based graph hypernetwork (GHN) processing the graph representation of architecture is trained to minimize the empirical risk of all clients. The output latent node features of GNNs are mapped back to the layer weights for training the original network. By converting a neural network into a graph and training it with a GHN model, heterogeneous model weights can be aggregated across FL clients indirectly by uploading local GHN weights to the FL server for aggregation.

B. Decentralized FedGNNs

As illustrated in Fig. 2(c), decentralized FedGNNs do not have a central server to coordinate FL clients. It only has one subscenario where the implicit graph is distributed in the clients. The implicit graph here only refers to the interclient graph.

1) Implicit Graph Distributed in the Clients: Since the interclient graph is distributed among the clients, it assumes that clients can only communicate with their neighbors. Thus, how to perform decentralized FL model aggregation is a challenging problem. Besides, how to communicate efficiently with their neighbors is also critical.

a) Decentralized FL aggregation problem: Existing works come up with two approaches to solve the decentralized FL aggregation problem where clients are related by a graph topology: 1) updating the FL model parameters via weighted summation of model updates within the neighborhood and 2) updating the FL model parameters via graph regularization.

In the weighted-summation-based approaches, FL clients communicate with their neighbors and update their local models by aggregating their neighbors' local model parameters based on the graph topology connecting them

$$\mathbf{w}_c^{r+1} = \sum_{j \in \mathcal{N}(c)} a_{cj} \cdot \left[\mathbf{w}_j^r\right] \tag{4}$$

where $\mathbf{w}_c^{r+1} \in \mathbb{R}^p$ denotes the local model parameters of client c at round $r+1$, which can be a Bayesian model [32], gated recurrent units [17], or GNNs [33]. $[\cdot]$ is the encryption operation for data privacy protection, such as Diffie–Hellman key exchange [33] or secret sharing [34]. a_{cj} is the [cth row, jth column] element in the adjacency matrix $\mathbf{A}$ of the graph, which is assumed to reflect the local data distribution similarity between client c and j. $\mathcal{N}(c)$ is neighborhood of c (including itself). All works under this section apply (4) once per round (i.e., a client only aggregates models from its one-hop neighbors). DSGT [35] utilizes the decentralized stochastic gradient tracking to achieve faster convergence.

In the graph-regularization-based approaches, graph Laplacian regularization is incorporated into the objective function to make model parameters from neighboring clients similar in order to address the non-i.i.d. problem [36]

$$R(\mathbf{W}, \mathbf{L}) = \mathrm{tr}\left(\mathbf{W}^T \mathbf{L} \mathbf{W}\right) = \frac{1}{2} \sum_{ij} a_{ij} \|\mathbf{w}_i - \mathbf{w}_j\|^2 \tag{5}$$

where $\mathbf{W} \in \mathbb{R}^{n \times p}$ denotes the model weights of neighboring clients. $\mathbf{L} \in \mathbb{R}^{n \times n}$ is the Laplacian matrix of the graph topology between neighboring clients. $\mathrm{tr}(\cdot)$ is the trace operation. a_{ij} is the edge weight in the adjacency matrix connecting client i and j. $\mathbf{w}_i \in \mathbb{R}^p$ indicates the model parameters in client i. Thus, for each client, the local objective function can be written as

$$\min_{\mathbf{w}_c} L_c(\mathbf{w}_c) + \lambda \cdot \frac{1}{2} \sum_{j \in \mathcal{N}(c)} a_{cj} \|\mathbf{w}_j - \mathbf{w}_c\|^2 \tag{6}$$

where λ indicates a balancing weight, $\mathcal{N}(c)$ represents the neighbors of client c, and $L_c(\cdot)$ is the local loss function in client c. Each client can only get the related information from their neighbors. Multitask learning is usually entangled with the decentralized FL aggregation problem, which can be solved with the above strategy. dFedU [37] assumes that each client has one task and a fully connected interclient graph is given in advance. Once each client obtains the locally updated models from its neighbors, it performs model updating with graph regularization. SpreadGNN [38] assumes that each client solves multiple tasks. An interclient task relationship graph is initialized from the task classifier model parameters. Clients apply decentralized periodic averaging stochastic gradient descent (SGD) (DPA-SGD) to optimize the objective function and update model weights and their corresponding task relationship graph iteratively with a convergence guarantee. Fed-ADMM [39] solves (6) by proposing a decentralized stochastic version of the alternating direction method of multipliers (ADMM) algorithm with rigorous statistical guarantees of their estimators.

b) Communication efficiency problem: Since the clients need to communicate with all their neighbors, communication efficiency is critical. Rizk and Sayed [34] propose a multiserver FedGNN architecture to increase communication efficiency when dealing with a large-scale graph. It assumes that there are multiple servers in the network related by a fixed graph topology and that there is no central server coordinating the network of servers. Clients under each server conduct FL model training following the classic centralized FL protocol. Once all the servers have aggregated their own clients' model updates, they perform interserver model aggregation following (4) among themselves. PSO-GFML [40] enhances communication efficiency by only exchanging a portion of local model parameters with the servers. Instead of knowing the adjacency matrix in advance, the graph can be learned during training, similar to graph attention networks (GATs) [41]. The edge weights of the interclient graph are calculated based on the similarity between unlabeled graph embeddings [42] or hidden parameters [43] in the corresponding clients.

C. Summary

In this section, we have discussed GNN-Assisted FL approaches that leverage the GNN model training to improve FL aggregation. We now summarize the techniques in terms of their advantages and disadvantages, as listed in Table I.

Centralized FedGNNs deal with the graph-structured FL system setting. With a GNN model trained in the FL server, it has more flexibility in the FL model aggregation to deal with the non-i.i.d. problem among FL clients. The GNN model plays a tradeoff between personalization on the client side and

TABLE I
SUMMARY OF ADVANTAGES AND DISADVANTAGES FOR VARIOUS GNN-ASSISTED FL SCENARIOS

Scenario	Sub-Scenario	Advantages	Disadvantages
Centralized FedGNNs	Implicit graph located in the server	• The server has more flexibility in FL aggregation to relieve non-IID problem.	• Difficult to prove the convergence. • The server requires high computation costs when the inter-client graph is large. • Imprecise inter-client graph deteriorates performance.
	Implicit graph located in the clients	• Relieve the non-IID problem between clients.	• Shared inter-client graph may leak privacy. • Imprecise inter-client graph deteriorates performance.
Decentralized FedGNNs	Implicit graph distributed in the clients	• Relieve the non-IID problem with personalized local models in clients. • Do not require a central server.	• High communication cost between clients. • Clients with higher centrality are vulnerable. • Need to re-train the model when new clients join.

generalization on the server side. However, it is more difficult to guarantee convergence with two objective functions (one for local models and the other for the global GNN model). Besides, when it is applied to an FL system with a large-scale interclient graph, the training cost of the GNN model in the server becomes enormous. In addition, an imprecise interclient graph in GNNs may deteriorate model performance. With the interclient graph stored by the clients with the GNN model trained locally, it can also relieve the non-i.i.d. problem. However, the locally stored interclient graph may leak other clients' private information. Besides, it also faces the same issue as above when the interclient graph is imprecise.

Decentralized FedGNNs are designed for the serverless graph-structured FL setting. It can address the non-i.i.d. problem among FL clients without a central server by making clients communicate with their neighbors directly. Due to the difference in the neighborhood for each client, the eventual aggregated model for each client is a personalized local model. However, such peer-to-peer learning with model weights shared directly between neighboring clients may result in high communication costs. Besides, clients with higher centralities are vulnerable to attacks. In addition, it is not continual learning as the local models need to be retrained when a new client joins as their neighbor.

IV. FL-ASSISTED GNNS

In this section, we introduce *FL-assisted GNNs* in the main taxonomy, dealing with the FL systems with explicit graph information, especially isolated graph data. Due to the isolation of graph data, FL algorithms, as an emerging tool to process distributed data with privacy protection, are applied to assist the GNN model training process.

According to the overlapping scale of graph node IDs across clients, *FL-assisted GNNs* have two scenarios: 1) *horizontal FedGNNs* and 2) *vertical FedGNNs*. For both scenarios, two more subscenarios are categorized separately based on the graph data properties in the clients: 1) clients without missing edges; 2) clients with missing edges for *horizontal FedGNNs*; 3) clients with incomplete graph data; and 4) clients with complete graph data for *vertical FedGNNs* (refer to Fig. 3). For each subscenario, we first introduce its setting assumption and the general strategy. Then, the main challenges are summarized with bold titles followed by detailed solutions. At the end of this section, the advantages and disadvantages are discussed for all subscenarios.

A. Horizontal FedGNNs

Horizontal FedGNNs refer to the situation, whereby the isolated graph data in clients share the same node feature spaces but different node IDs. Each client has at least one graph or a set of graphs. Depending on the loss of edges connecting nodes in different clients, clients can have local graph data without missing edges [see Fig. 3(a)] or with missing edges [see Fig. 3(b)].

1) Clients Without Missing Edges: This setting assumes that edges connecting nodes in different clients are retained by clients [the left figure in Fig. 3(a)] or no such edges exist between clients [the right figure in Fig. 3(a)]. The common strategy for clients without missing edges is to train local GNN models in the clients to learn the local graph representations or node embedding first. Then, an FL algorithm is applied on top of it. The FL server collects the model parameters or gradients from clients for FL aggregation, as described in (2) [44], [45], and sends back the updated parameters to the clients for the next round of training. FedGNN works under this setting are introduced according to the research problems that they solve. GNN-related research problems focus on non-i.i.d. problems, graph embedding problems with distributed graph data, and neural architecture search problems with distributed graph data. FL-related research problems focus on the communication efficiency problem and privacy protection problems between clients.

a) Non-i.i.d. problem in distributed graph data: Graph data consist of data and graphs. The non-i.i.d. problem exists in both data distribution and graph topology [46]. The non-i.i.d. problem in data distribution, including attributes distribution and label distribution, is the same as the one for normal data. The non-i.i.d. problem in graph topology refers to the non-i.i.d. distributions of node degrees, edge weights, edge types, or graph structures among clients. Most existing FedGNN works focus on the former one, providing some personalized FL inspired solutions [16]: model-based approaches, data-based approaches, and

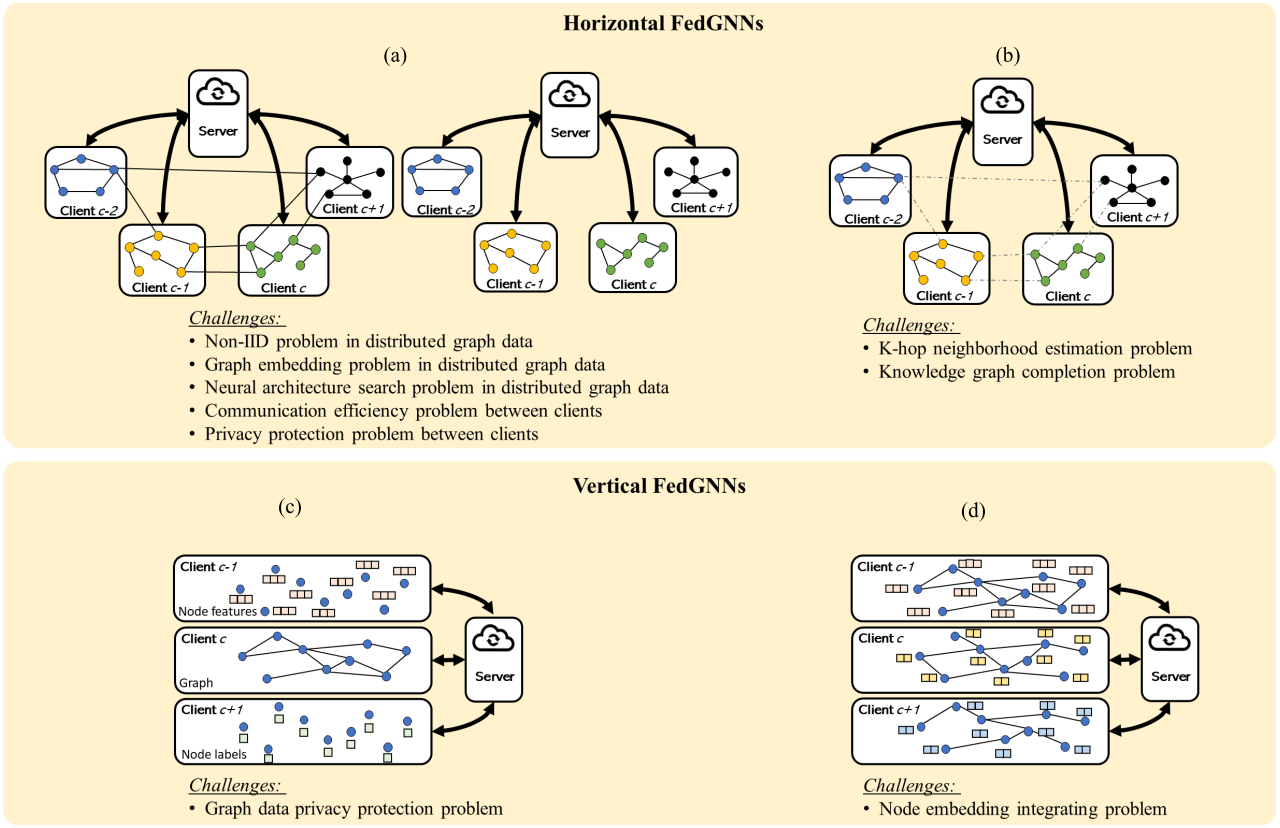

Fig. 3. Illustrations of subscenarios with corresponding challenges in the FL-assisted GNN category. (a) Clients without missing edges. (b) Clients with missing edges. (c) Clients with incomplete graph data. (d) Clients with complete graph data.

FL-aggregation-improvement-based approaches. A few works put some lights on the latter recently.

Model-based approaches improve the adaptation performance of the local model or learn a powerful global FL model for the future personalization of each client, which includes model interpolation, regularized local loss, metalearning, knowledge distillation, and so on. ASFGNN [47] and FedEgo [48] apply the model interpolation technique in the client. The final model for the client is a combination of the global model and the local model. The percent of the local model in the updating process is controlled by a mixing weight, which can be the Jensen–Shannon divergence [47] or the earth mover distance (EMD) [48] between local and global data distributions. FedAlign [49] adds an optimal transport (OT) distance-based regularization term between local and global models in the loss function to minimize the model divergence as FedProx [50]. GraphFL [51] applies a metalearning training scheme to mitigate the non-i.i.d. problem. Inspired by the model-agnostic metalearning (MAML) [52], it finds a good initial model that can be fast adapted to clients after a few local updates. InfoFedSage [46] utilizes a generative module in the server and an information bottleneck regularizer in the clients to alleviate the non-i.i.d. problem. FedGraph-KD [53] and FGSSL [54] apply knowledge distillation to transform the global information from the server and, meanwhile, preserve the local information to solve the non-i.i.d. problem.

Data-based approaches target to decrease the statistical heterogeneity of client data distributions using sample reweighting, clustering, manifold learning, and so on. FLIT [55] solves the non-i.i.d. data problem by reweighting samples based on their prediction confidence. To make local training more consistent across clients and avoid overfitting the local data, it puts more weight on the samples that the local model is less confident in the prediction results than the global model. GCFL+ [56] and FedINT [57] solve the non-i.i.d. problem by clustering clients based on the gradients or model weights of the graph isomorphism network (GIN) model [58] from each client. General FL aggregation is applied within each cluster. FMTGL [59] relieves the non-i.i.d. problem by obtaining universal task representation with a fusion module shared across clients. The key to getting the universal task representation is to process the multisource representation matrices on a common fusion space, consisting of several learnable support vectors.

FL-aggregation-improvement-based approaches relieve the non-i.i.d. problem by defining the importance of each client in FL aggregation. The general FL aggregation approach is described as follows:

$$\mathbf{w}^r = \sum_{c=1}^{C} \eta_c \cdot \mathbf{w}_c^r \tag{7}$$

where $\mathbf{w}_c^r$ and $\mathbf{w}^r$ denote the model weights in the cth client and the global model weights in the server at the round r. η_c represents the importance of client c in FL aggregation. η_c can be defined by many factors. Apart from the number of samples in the clients, Fed-CBT [60] defines it with the training round index. It puts a higher weight on the clients with the latest updates. In [61], the weight is defined by a trainable attention mechanism based on the global model parameters and local model parameters layerwisely. Fed-Pub

[62] calculates the weights based on the functional similarity between the client's local model outputs of a public randomly generated graph. To further solve the non-i.i.d. problem, each client calculates the similarities with all other clients to get a personalized local model.

Some recent works focus on the non-i.i.d. problem in the graph structure. FedLIT [63] employs k-means clustering to categorize edges by type. It then utilizes multiple convolution channels in each client for message passing on different types of edges, resolving the non-i.i.d. challenge. FedStar [64] adopts a feature-structure decoupled GNN with distinct feature-based and structure-based encoders. For FL aggregation, only the structure-based encoder is uploaded to the server, enabling the sharing of structural knowledge across non-i.i.d. graphs from diverse domains.

b) Graph embedding problem in distributed graph data: As the local graph topology changes over time or each node contains time-series data, it is necessary to embed the spatial–temporal (ST) information for the distributed graph data. STFL [65], Feddy [66], and 4D-FED-GNN+ [67], [68] deal with ST graph data embedding differently. STFL [65], [68] does not consider the temporal relationship in the graph embedding. They treat the graph data at each time step as one sample and predict the label for each of them. Feddy [66] considers temporal information in the graph embedding by applying dynamic GNNs. 4D-FED-GNN+ [67] focuses on the evolution graph learning task with missing time points. Each client trains a GNN model for each time step. GNN acts as a generator or a self-encoder based on the data availability at the current and next time step. It improves the predictive performance of local models while benefiting from other clients with data corresponding to the missing time points.

c) Neural architecture search problem in distributed graph data: Designing a suitable architecture for the local GNN model is beneficial for the global model's performance. Existing works pay attention to the GNNs architecture search problem. FL-AGNNS [69] proposes a federated evolutionary optimization strategy to search for suitable GCN architectures. It applies a GCN SuperNet and a weight-sharing strategy to reduce the searching time so that the proposed algorithm can find better GCN models in a short time. The approach in [70] searches for the GNN models that can produce the most reproducible features. The model is selected based on a reproducibility matrix between paired GNN models. The reproducibility value between the two GNNs is defined by the overlap ratio of the top K reproducible features, which are selected according to the last layer of the models.

d) Communication efficiency problem between clients: Large-scale local graph data require local GNN models with an enormous number of parameters, which results in high communication costs between clients. To reduce the communication and computation cost of GNN model training, FedGraph [71] designs a sampling policy for the server based on reinforcement learning. In each round, the server refines the sampling strategies (i.e., the number of nodes to be sampled) and the GNN model parameters for the clients. CTFL [72] clusters clients based on the closeness of their local model parameters with a divide-and-conquer strategy. Only one representative local model of each cluster is sent to the server for FL aggregation to reduce communication costs.

e) Privacy protection problem between clients: By sharing model parameters and graph topology, FedGNNs have large attack surfaces, requiring more attention to privacy protection. Some works assume that both clients and the server are honest-but-curious. FeSoG [73] uploads encrypted gradients using dynamic local differential privacy (LDP) to the server for FL aggregation. Clients in FedGraph [71] encrypt local training with a trusted execution environment (TEE) and the server encrypts global model aggregation with secure multiparty computation (MPC) or homomorphic encryption (HE). FedEgo [48] protects the graph privacy by constructing mashed ego-graphs in the client. The global structure is protected by sampling neighboring nodes with a fixed size for the central node to construct the ego-graphs in the clients. Local graph embedding is anonymous by averaging a batch of ego-graphs (mixup or mashed) before being uploaded to the server. SOS [74] encrypts the original graph by generating new graphs from the original ones based on the information-bottleneck principle and then employs the local GNN models for training on these generated new graphs. FDL-TF [75] ensures privacy by introducing a trusted authority to verify the accuracy of uploaded model weights before FL aggregation in the server. Some assume that only the server is honest-but-curious and clients are honest. For example, ESA-FedGNN [76] utilizes Shamir secret sharing and double-mask strategy to ensure privacy and prevent malicious adversaries from stealing model parameters.

2) Clients With Missing Edges: This setting assumes that some edges connecting nodes in different clients are missing [refer to Fig. 3(b)]. The missing edges can be divided into two types: 1) edges between nodes with different node IDs and 2) edges between aligned nodes in different clients. For the first type, how to deal with the k-hop neighbor information isolated by the missing edges is a key challenge. Since local clients cannot communicate with each other directly, amending the local graph with estimated edges to mimic the k-hop neighbor information in other clients is a common strategy. A complete local graph can ensure that a high-quality graph representation and edges between clients can mitigate the non-i.i.d. data problem to some extent. For the second type, KG completion is an important challenge in this setting. The key strategy is to transform information between aligned nodes across clients to help local KG embedding completion. Once the local graphs are amended, the FL algorithm is applied to assist GNN training in the same way as the "client with no missing edge" works.

a) K-hop neighborhood estimation problem: Edge generators or node generators, located in the server, clients, or a third party, are designed to amend the local graphs to estimate their k-hop neighborhood information. Once the local graphs are complete, GNN models are trained on them in the clients, and an FL aggregation is applied in the server to obtain a global model.

FASTGNN [77] proposes a simple edge generator in the server. It reconstructs the missing edges between clients with Gaussian randomly generated edges and broadcasts them to

all clients to update their local graphs. The edge generator in FedGL [78], located in the server, can generate a global pseudograph with node embeddings uploaded by FL clients and distribute it to the clients to amend their local graphs for GNN model training.

Instead of generating missing edges directly, some works [79], [80] designed generative models to recover missing neighborhood node embedding first and then reconstruct missing edges based on them. FedSage+ [79] proposes a node feature generator in the client. To train the generator, the client holds out some existing edges randomly in the local graphs. The generator, equipped with a Gaussian noise generator, is trained to predict the number of missing neighborhood nodes and reconstruct hold-out neighborhood node features. Once the subgraph is updated, each client trains a GNN model continually, GraphSage [4], and uploads the model parameters to the server for FL aggregation. FedNI [80] improves the node generator by adding a discriminator to identify if the node features come from the generator or the real missing neighbor. This generative adversarial network (GAN)-based generator [81] improves the quality of generated node features. Besides, it removes nodes in the client with a breadth-first search, instead of random selection.

DP-FedRec [82] leverages a private set intersection (PSI) to extend the local graph and relieve the non-i.i.d. problem. Clients i and j execute the PSI protocol to get the intersected vertex. The client i extends the edges and vertex for the intersected vertex within k-hop from the client j. To protect privacy, all clients add noise to the graph data before the local graph extension step, which is referred to as applying differential privacy (DP).

FedPerGNN [83] and FedGNN [84] amend their local graphs in a more safe way. They introduce a third-party server, which only deals with graph expansion for the clients. The original central server first generates and sends a public key to clients for local node IDs and embedding encryption. Then, clients upload the ciphertexts to the third-party server. The third-party server locates the interacted nodes by checking the ciphertexts of their IDs and distributes encrypted node embedding to the clients to amend their local graphs for the following local GNN training.

Some works [85], [86] pay more attention to communication efficiency with simple graph amending methods. In [85], the client sends a request via the central server to the corresponding clients to get the missing embedding by neighbor sampling. It proposes an algorithm that can find an optimal sampling interval that achieves the best tradeoff between convergence and running time. FedGCN [86] allows clients to collect one- or two-hop averaged neighbor node features from other clients once at the beginning of the training to amend missing information. Then, each client trains the local GNN model, and the server collects local model weights for FedAvg-based FL aggregation. FedGCN also provides a theoretical analysis of the tradeoff between the convergence rate and communication cost under different data distributions.

b) KG completion problem: KGs from different domains may contain the same entities. How to improve KG embedding quality with the help of other KGs without leaking privacy is a challenge. The key part is to transform information between aligned embedding across clients.

FKGE [87] designs a revised GAN-based module [88] to translate the aligned entity and relation embedding between paired KGs. If the paired KGs are improved, the refined embedding is broadcast to other KGs. FedE [89] designs an overall entity table in the server to record all unique entities from clients. The server applies FedAvg FL aggregation on the aligned entity embedding in the table. Once finished, the updated entity embeddings are distributed to clients. The clients update entity embedding based on KG embedding methods with a self-training contrastive learning loss.

To tackle the privacy leakage issue in FedE, clients in FedR [90] upload relation embeddings instead of entity embeddings since the server cannot infer entity embedding given only relation embedding. To further protect the privacy, secure aggregation [91] is applied to the relation embedding before being uploaded to the server, and the relation table in the server is obtained via private set union (PSU).

FedEC [92] improves FedE with the non-i.i.d. problem by adding a regularization term in the loss. It can increase the similarity between global and local entity embedding in the current round and decrease the similarity between local entity embedding in the current and the last round.

B. Vertical FedGNNs

Clients in *vertical FedGNNs* hold nodes with completely overlapping node IDs but different feature spaces. Clients train a global GNN model with features from different clients with the help of FL. According to the completeness of graph data, *vertical FedGNNs* have two subscenarios: 1) clients with incomplete graph data and 2) clients with complete graph data.

1) Clients With Incomplete Graph Data: This setting assumes that graph topology, node features, and node labels are owned by different clients. That is, clients do not have complete graph data (neither node features nor graph topology) [see Fig. 3(c)]. For example, in an FL system with three clients, one client owns the node features, one owns the graph topology, and one owns the node labels, or one gets node features, and the other owns the rest if there are only two clients in the system. How to make these clients work together while protecting their privacy is a key challenge.

a) Graph data privacy protection problem: Instead of sharing the original adjacency matrix of the graph, SGNN [93] calculates a dynamic time warping (DTW) algorithm-based similarity matrix to convey the same graph topology but conceal the original structure. To protect the privacy of the node features, one-hot encoding is applied to map the original features to a matrix. Then, the information from different clients is uploaded to the server to train a global GNN model for a node classification task. FedSGC [94] assumes that there are only two clients without a central server. Graph topology and node features are owned by two clients. The client who has the node labels is the active party to create encryption key pairs. Clients encrypt the sensitive information using additively HE (AHE) before sending them to the other party for the GNN model parameter updating.

TABLE II
SUMMARY OF ADVANTAGES AND DISADVANTAGES FOR DIFFERENT SCENARIOS OF FL-ASSISTED GNNS

Scenario	Sub-Scenario	Advantages	Disadvantages
Horizontal FedGNNs	Clients without missing edges	• Train GNN models with isolated graph data. • Relieve the data heterogeneity across clients.	• Biased graphs may cause unfairness. • Few works encrypt the model weights. • Vulnerable to malicious attacks. • High communication costs. • The framework only works with limited GNN models.
	Clients with missing edges	• Train GNN models with isolated graph data. • Mitigate the impact of missing information across clients.	• Imprecise amended local graph deteriorates performance. • Shared node feature may leak privacy when amending local graph. • Others are the same as above.
Vertical FedGNNs	Clients with incomplete graph data	• Train a GNN model with isolated graph data.	• Number of clients is limited. • Vulnerable to malicious attacks. • The framework only works with limited GNN models.
	Clients with complete graph data	• Train a GNN model with isolated graph data.	• Vulnerable to malicious attacks. • The framework only works with limited GNN models.

2) Clients With Complete Graph Data: In this setting, it assumes that clients contain complete graph data, including graph topology and node features. However, their node feature types are different [refer to Fig. 3(d)]. Integrating the node features in a safe way is the key strategy to train the GNN model with such distributed graph data.

a) Node embedding integrating problem: Clients in VFGNN [95] integrate DP encrypted node features in the semihonest server via mean, concatenation, or regression calculations as FL aggregation. Once training is complete, the client who owns node labels receives the updated node embedding from the server to perform node prediction. FML-ST [96] assumes that there is a global pattern graph shared by all clients with the same nodes. They fuse the local ST pattern and global ST pattern using a multilayer perceptron (MLP) with concatenated patterns as inputs. Clients leverage the global pattern to personalize their local pattern graph by evaluating the difference between global and local pattern graphs. FedVGCN [97] transfers homomorphic encrypted intermediate results between two clients to complete node feature space with the semihonest server creating encryption key pairs and doing FL aggregation. Graph-Fraudster [98] studies the adversarial attacks on the local raw data and node embedding. It proves that the DP mechanism and top-k mechanism are two possible defenses to the attacks.

C. Summary

In this section, we have discussed FL-assisted GNN approaches that leverage FL to assist GNN model training in a distributed setting. We now summarize the techniques in terms of their advantages and disadvantages, as listed in Table II.

Horizontal FedGNNs deal with clients having graph data with different node IDs. They can train GNN models with isolated graph data and relieve the data heterogeneity problem across clients. However, they generally overlook the heterogeneity in the graph topology. Biased graphs may cause unfairness in FedGNNs. Besides, the privacy protection capability achieved by existing works is generally low. For example, few works consider encrypting the model weights before sending them to the server, making current works lack robustness against malicious attacks. In addition, communication costs are high when the local graph model size or the number of clients is large. Finally, existing approaches only work with limited basic GNN models. More advanced GNN models need to be included. Approaches capable of dealing with more difficult situations where edges between clients are missing can recover some missing information to improve performance. However, if the amended local graph is imprecise, the model performance may deteriorate. Besides, some approaches require clients to share some node features with the neighbors to repair local graphs, which can cause privacy leakage.

Vertical FedGNNs deal with clients having graph data of different node features but the same node IDs. They can help clients train a GNN model with isolated graph data. However, for the incomplete graph data setting, existing approaches only support two clients (pairwise FL) or three clients, which is insufficient. Besides, these systems are vulnerable to malicious attacks, and they only work with limited basic GNN models.

V. AUXILIARY TAXONOMY

In this section, we discuss the proposed auxiliary taxonomy for FedGNNs. According to the level of heterogeneity of local data and models across FL clients, existing works can be divided into three categories (with increasing heterogeneity levels): 1) clients with the same nodes; 2) clients with different nodes but the same network structure; and 3) clients with different network structures. To implement FL aggregation under different situations, diverse strategies have been proposed with various intermediate information exchanged among FL clients (see Table III).

A. FL Clients With the Same Nodes

In this category, it is assumed that clients have nodes with the same set of nodes but different types of node features.

TABLE III
SUMMARY OF FEDGNN LITERATURE IN THE 2-D TAXONOMY

			Same Nodes	Different Nodes, Same Network	Different Networks
GNN-Assisted FL	Centralized FedGNNs	Implicit graph located in the server	nil	CNFGNN [29], MLFGL [23], HFME [28], Big-Fed [24], SFL [25], PDGNet [26], [27]	nil
		Implicit graph located in the clients	nil	FedCG [30]	HAFL-GHN [31]
	Decentralized FedGNNs	Implicit graph distributed in the clients	nil	D-FedGNN [33], DSGT [35], [32], [17], [34], PSO-GFML [40], FedSTN [43], SemiGraphFL [42], Fed-ADMM [39], dFedU [37], SpreadGNN [38]	nil
FL-Assisted GNNs	Horizontal FedGNNs	Clients without missing edges	nil	Fed-CBT [60], STFL [65], [70] 4D-FED-GNN+ [67], FedGCN [61], FedGraph [71], Feddy [66], FeSoG [73], FL-AGNNS [69], ASFGNN [47], InfoFedSage [46], FLIT/FLIT+ [55], GCFL/GCFL+ [56], Fed-RGCN [49], GraphFL [51], FedEgo [48] GraphSniffer [44], DA-MRG [45], CTFL [72], FMTGL [59], ESA-FedGNN [76], Fed-Pub [62], FedGraph-KD [53], FGSSL [54], FedLIT [63], FedStar [64], [68], SOS [74], FDL-TF [75], FedINT [57]	nil
		Clients with missing edges	FedE [89], FedR [90], FedEC [92], FKGE [87], FedGL [78]	FedGL [78], FASTGNN [77], FedNI [80], FedSage/FedSage+ [79], FedPerGNN [83], FedGNN [84], DP-FedRec [82], [85], FedGCN [86]	nil
	Vertical FedGNNs	Clients with incomplete graph data	FedSGC [94], SGNN [93]	nil	nil
		Clients with complete graph data	Graph-Fraudster [98], FedVGCN [97], VFGNN [95], FML-ST [96]	nil	nil

Vertical FedGNNs and part of horizontal FedGNNs with overlapping nodes (e.g., KG completion tasks) belong to this category. In this situation, FL clients usually upload the node feature embeddings to the FL server, where different feature embeddings of the same node are aggregated for knowledge transfer among clients. Besides, model weights can also be shared among clients to jointly train a global FL model in vertical FedGNNs.

B. FL Clients With Different Nodes but the Same Network Structure

In this category, it is assumed that FL clients have different nodes but with the same network architecture. Most GNN-assisted FL works and horizontal FedGNN works belong to this category as their clients contain different samples. Since the node IDs are different under this setting, node feature embedding cannot be uploaded to the server for FL aggregation. According to GNN aggregation $\mathbf{X}^{(l+1)} = \mathbf{A}\mathbf{X}^{(l)}\mathbf{W}$, the size of the trainable matrix $\mathbf{W}$ is not related to the graph topology but related to the dimensions of node features and output features. Thus, the embedding of the entire graph [43], [96], trainable model weights, and gradients in the GNN model can be uploaded to the server for FL aggregation. Whether to upload model weights or gradients depends on the preference between training speed and model performance [22]. Uploading model weights to the server allows each client to perform multiple epochs of local training. However, its local model update direction may deviate from the global FL model. Uploading gradients allows the client to closely follow the latest global optimization direction in every updating step. However, the frequent communication between clients and the server may increase communication costs during the training process.

C. FL Clients With Different Network Structures

Works in this category deal with a more difficult situation where network architectures in clients are different. GNN model weights from different clients share the same dimensions as long as the node feature dimensions are the same across clients. However, in this setting, the node features from different clients are heterogeneous, resulting in various local network architectures across clients. Currently, only Litany et al. [31] provide a solution. It converts the local network architecture into a graph with each layer as a vertex and the layer type as the node features. Once the number of node features is consistent across clients, it becomes the second category in Section V-B. A GNN model is trained with the graph, and GNN model weights are uploaded to the server for FL aggregation. However, the current solution is limited to a few predefined network architectures, which is not efficient and flexible.

VI. IMPLEMENTATION

Performance benchmarking is an essential factor for the long-term improvement of the FedGNN research field. In this section, we review and discuss the applications with benchmarks, evaluation metrics, experiment evaluation designs, and platforms in the existing FedGNN literature.

A. Applications

There are several benchmark datasets developed for GNNs, including citation network datasets, social network datasets, and chemical property datasets. FedGNNs test their algorithms on these datasets [17], [24], [33], [33], [38], [47], [49], [51], [53], [54], [55], [55], [56], [56], [61], [62], [63], [64], [71], [74], [78], [79], [94], [95], [97], [98], [99] with various data partition methods. FedGNNs also explore many GNN applications in a decentralized setting with privacy concerns. FedGNNs have been applied in KG completion [87], [89], [90], [92] and recommendation system tasks [73], [83], [84] with privacy protection by considering one KG or one user as one client. Besides, income prediction [27], malicious transaction detection [44], [45], network anomaly detection [68],

and Internet-of-Things (IoT) threat detection [57] are also potential applications for FedGNNs. FedGNNs have been used in FL applications with graph-structured information. In computer vision applications, FedGNNs can improve the image classification performance [30], [32], [37] by making close clients have similar local models. It can also help the image classification training with heterogeneous network structures across clients by converting the local neural network structure into a graph [31]. Healthcare applications [60], [65], [80] can be addressed with FedGNNs when the data contain graph structures or there are relationships among patients. For example, brain imaging data can be parceled into different regions of interest (ROIs) with each ROI as one node in the graph. The population graph between patients can improve disease prediction with the help of FedGNNs [60]. Transportation can leverage FedGNNs in many situations, such as traffic flow prediction [29], [43], [77], [96], object position prediction [66], or indoor localization [23], [100]. Sensors or surveillance cameras can be modeled as clients connected by a map graph. Alternatively, the objects detected by the cameras can form a local graph.

B. Evaluation Metrics

The usage of evaluation metrics depends on the learning task. In general, there are two types of tasks: 1) classification task and 2) regression task. For classification tasks (e.g., node classification, graph classification, and image classification), accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (ROC-AUC) have been adopted as evaluation metrics. For regression tasks (e.g., node embedding regression, graph embedding regression, graph learning, and link prediction), the following evaluation metrics are adopted.

1) For node and graph embedding regression tasks (e.g., recommendation system [73] and traffic flow prediction [77]), mean absolute error (MAE), mean square error (mse), root mse (RMSE), and mean absolute percentage error (MAPE) are adopted as the evaluation metrics [55], [66], [73], [82] to measure the distance between predicted values and the ground truth.
2) For graph learning tasks (e.g., brain connectivity estimation [60]), apart from the MAE, the graph learning performance can be measured by the Frobenius distance between the estimated graphs and the ground truth.
3) For link prediction tasks (e.g., KG completion [87], [89], [90], [92]), the link prediction performance can be evaluated with mean rank, mean reciprocal rank (MRR), and the proportion of correct entities in top N ranked entities (Hits@N).

FedGNN applications, the corresponding datasets, research problems, and evaluation metrics are summarized in Table IV.

C. FedGNN Experimental Evaluation Design

Since there are few real cross-silo graph datasets, the majority of works simulate the distributed setting by performing partitioning on the public datasets summarized in Table IV. Here, we discuss the experimental evaluation design in the FedGNN literature according to the different scenarios in the main taxonomy.

In the GNN-assisted FL setting, an interclient graph exists between clients with various local data types, such as graph data, image data, or temporal data. For the local data partition, clients with i.i.d. and non-i.i.d. data distribution have different data partition methods. Besides, there are several ways to build an interclient graph. These methods are summarized as follows.

To construct FL clients with the i.i.d. data distribution, samples are distributed evenly and randomly to clients. A sample can be an image [31], [32] (e.g., an image from MNIST), a time-sequence data from a sensor [23], or a protein graph from some biomedicine dataset [42]. A sample can also be a node in the graph [17] with a different partition method: each client selects some seed nodes randomly and expands the graph with a breadth-first search on the original entire graph to get their local data.

To construct FL clients with non-i.i.d. data distributions, there are five main approaches.

1) *Imbalanced Partition:* Different clients have different numbers of samples. It can be achieved by distributing the samples using a latent Dirichlet allocation (LDA) [33], [38].
2) *Clustering Partition:* Different clients have different clusters. It can be achieved by dividing the samples into clusters using a clustering algorithm (e.g., k-means) [23], [42].
3) *Label Distribution Skew Partition:* Different clients have different subsets of label classes. It can be achieved by making one client maintain data only from one class or a subset of classes [27], [30].
4) *Natural Identity Partition:* Different clients have different characteristics. For example, in the traffic flow dataset, one sensor is assigned to one client [29]. In human activity recognition, one person with his/her data is considered as one client [37]. In nature language processing (NLP) datasets, documents from one domain are assigned to one client [24].
5) *Synthetic Data Generation:* Generate synthetic data points for clients following Gaussian distribution with different variances [34], [35], [40].

The interclient graphs can be obtained as follows.

1) *Natural Graph:* Some datasets contain a natural graph topology that can be used as an interclient graph directly. For example, in traffic flow data, the road map can be used as an interclient graph with each sensor as one client [29]. In the NLP dataset application, the syntactic structure can also work as the graph between different domains as clients [24]. The wireless communication network can also serve as it between routers [26].
2) *Simulated Graph Based on Clients' Similarities:* Some datasets have no natural graph topology. The interclient graph can be calculated based on the clients' information, such as clients' embedding [23], [27], [27], [28], [30], [37].

TABLE IV
SUMMARY OF APPLICATIONS, DATASETS, AND CORRESPONDING EVALUATION METRICS IN FEDGNNS

Applications		Datasets	References	Research Problems	Evaluation Metrics
Citation Network		Citeseer, Cora, PubMed, Cora-Full, Physics, Coauthor CS, MSAcademic, Amazon2M, ACM, AIFB, BGS, Wiki, Aminer, ogbn-arxiv, DBLP	[17], [71], [99], [47], [51], [78], [79], [97], [98], [46], [94], [95], [54], [62], [63], [74]	Node classification	Accuracy
Social Network		REDDIT, REDDIT-BINARY, COLLAB, IMDB_BINARY, IMDB_MULTI, GITHUB_STARGAZERS, Zachary karate club network NEGAME	[24], [56], [53], [64]	Node classification	Accuracy
Chemical property prediction		FreeSolv,QM9, Lipophilicity, ESOL	[33], [55]	Graph embedding regression	MAE, MSE, RMSE
		Tox21,SIDER,BACE, ClinTox, BBBP, ENZYMES, D&D, PROTEINS, PPIN, MUTAG, NCI1	[33], [38], [61], [49], [55], [56], [64]	Graph classification	Accuracy, F1-score, ROC-AUC
Knowledge graph completion		FB15k-237, NELL-995, WN18RR, DDB14, Dbpedia, Yago	[89], [90], [92], [87]	Link prediction	MRR, Hits@N
Recommendation		Ciao, Epinions, Filmtrust, Flixser, Douban, YahooMusic, MovieLens(ML)-100K/1M/10M	[73], [83], [84]	Node embedding regression	MAE, MSE, RMSE
Income Prediction		UCI dataset (Adult)	[27]		
Malicious Transaction detection		Elliptic Data Set, Twi-Bot-20	[44], [45]	Node classification	F1-score
Network anomaly detection		Ford Transit 500, CAN Signal Extraction and Translation Dataset	[68]	Graph classification	Accuracy, precision, recall, F1-score
IoT threat detection		IFTTT, SmartThings, Alexa platform dataset	[57]	Graph classification	Accuracy, precision, recall, F1-score
Computer Vision	Image classification	CelebA, MNIST, MedMNIST, Federated Extended MNIST, CIFAR-10, CIFAR-100 Chest X-ray, Vehicle sensor, Human Activity Recognition, Letter-low, Letter-med, Letter-high	[28], [30]–[32], [37], [63]	Classification	Accuracy
Healthcare	Brain template estimation	ABIDE-I, OASIS-2	[60]	Graph learning	Frobenius distance, MAE
	Sleeping stage classification	ISRUC_S3	[65]	Classification	Accuracy, F1-score
	Disease prediction	ADNI, EHRs	[80], [63]	Node classification	Accuracy, ROC-AUC
Transportation	Traffic flow prediction	PEMS-BAY, METR-LA, TaxtNYC, TaxtBJ, PeMSD4, PeMSD7, Citi-Bike Dataset from New York City, Citi-Bike Dataset from Washington DC, Citi-Bike Dataset from Chicago, Beijing Metro Dataset	[29], [43], [96], [77], [75]	Node embeding regression	MAE, RMSE, MAPE
	Object position prediction	Stanford Drone Dataset (SDD)	[66]		
	Indoor localization	Received signal strenth (RSS) dataset of a shopping mall	[23], [100]		
	Airport busyness prediction	Brazilian air-traffic network European air-traffic network	[93]	Node classification	Accuracy

3) *Synthetic Graph:* A fully connected graph [17], [42], ring connected graph [32], or just a randomly generated graph can be used as the interclient graph [34], [35], [40].

In the FL-assisted GNN setting, clients maintain a set of graphs or a set of nodes (one graph). We summarize the main data partition methods for both horizontal and vertical FedGNNs in the following.

In the horizontal FedGNN scenario, clients have different node IDs. If each client has a set of graphs, the data partition methods for i.i.d. and non-i.i.d. are very similar to the GNN-assisted FL setting. For i.i.d., graph samples are distributed to clients evenly and randomly [60], [61], [66], [67], [70], [80]. For non-i.i.d., imbalanced partition [55], label distribution skew partition (graphs of one class are assigned to one client) [65] and natural identity partition (one dataset is assigned to one client) [56] are applied.

If each client has only one graph, it is more complex for both settings (i.i.d. and non-i.i.d.) to partition one entire graph into several subgraphs. For i.i.d., two methods are applied: 1) nonoverlapping partition by randomly dividing the entire graph into several subgraphs [77] and 2) overlapping partition by randomly assigning a portion of nodes or edges to one client [51], [71], [78], [89], [90], [92]. For non-i.i.d., strategies used in GNN-assisted FL settings still work albeit with some changes. For clustering partition, nodes, instead of general samples, are grouped using various methods, such as k-means, METIS [101], and Louvain [102] for graph data. These groups are then assigned to clients [54], [62], [72], [74], [79], [99]. For label distribution skew partition, each client selects most nodes from major classes and few nodes from minor classes [47], [48], [85]. For natural identity partition, more identity units are applied to partition the data. In the recommendation application, one user with its interactions is

considered as one client [73], [83], [84]. In the KG completion application, one KG is considered as one client [87]. In some citation network applications, papers published in the same year are assigned to one client [59], [85].

In the vertical FedGNN scenario, clients have the same node IDs. For the incomplete graph setting, if there are three clients, then each of them gets one of the node features, graph topology, and node labels of graph data [93]. If there are only two clients, then one will get one component of graph data and the other gets the rest [94]. For the complete graph setting, node features are divided evenly to clients with the graph topology retained by all clients [95], [96], [97], [98].

D. FedGNN Platforms

Currently, there are two FedGNN platforms. FedGraphNN [19] is an open-source platform supporting three GNN models and two FL aggregation methods. It has collected 36 graph datasets and partitioned them into distributed silos, forming a promising FedGNN benchmarking tool. FederatedScope-GNN [103] consists of an event-driven FedGNN framework with two components: 1) ModelZoo and 2) DataZoo. ModelZoo provides comprehensive GNN models (e.g., GCN [104], GAT [41], and GraphSage [4]) used in the clients and some of the existing FedGNN models (e.g., FedSage+, FedGNN, and GCFL+). DataZoo provides a collection of splitting strategies for distributing a given graph dataset among FL clients.

VII. Promising Future Research Directions

As an emerging field, FedGNN research is starting to gain traction. Nevertheless, for this technology to effectively deal with challenges in real-world applications, many problems remain to be addressed. Here, we highlight seven of them, which hold promising opportunities.

1) *Robust FedGNNs Against Malicious Attacks:* By sharing node embeddings, graph topology, and model parameters, FedGNNs have large attack surfaces. Although some works attempt to address this issue by leveraging DP [77], [84], [87] or cryptographic methods [34], [47], [71], [95], they are designed to guard against only semihonest attackers. Additional research is needed to explore how FedGNNs can be made more robust in the face of malicious privacy attacks.
2) *Explainable FedGNNs to Improve Interpretability:* Works on the explainability of GNNs [105] and FL [106] are starting to emerge. FedGNN involves complex model structures and training processes. Thus, achieving explainability [107] under this setting is even more challenging. The incorporation of explainability into FedGNNs needs to jointly consider the needs for interpretability by the stakeholders involved while balancing the goals of preserving privacy and training models efficiently.
3) *Efficient FedGNNs for Large-Scale Graph Data:* Existing FedGNNs are generally studied with small-scale distributed datasets. Thus, communication efficiency has not yet been adequately considered. However, in order to scale FedGNNs up to large-scale graph data (e.g., KGs), communication overhead can be an important bottleneck since the clients often adopt multilayer GNN models with a large number of model parameters to be transmitted.
4) *Fair FedGNNs for Clients With Biased Graphs:* Graph data consist of the graph topology and data. Existing FedGNNs mainly focus on the non-i.i.d. problem in data distribution while ignoring the non-i.i.d. problem in graph topology distribution across FL clients. Different clients may own graphs with different properties. For example, some clients have graphs from one class with high node degrees, while others have graphs with low node degrees [108], [109]. Such biased graphs can affect outcomes or even cause harm in the FedGNNs [110]. Thus, achieving fairness in this setting is important.
5) *Continual FedGNN Training for New Clients:* In GNN-assisted FL, clients are treated as nodes in the graph. As new clients join, a well-trained global model can be applied to them directly. However, in the decentralized setting, since there is no server to coordinate the global training, all clients need to retrain their local models when new clients join. Thus, to improve the training efficiency, it is necessary to develop continual learning algorithms for decentralized FedGNNs.
6) *Comprehensive FedGNN Frameworks Supporting Diverse GNN Models:* Various GNN algorithms have been rapidly emerging in recent years. However, existing FedGNNs only employ a limited set of GNN models (e.g., GCN, GAT, and GraphSage). Thus, to make full use of GNNs to assist FedGNNs in solving more difficult problems, comprehensive FedGNN frameworks with more GNN algorithms or strategies are required.
7) *Realistic Cross-Silo Graph Datasets for Benchmarking:* Existing FedGNNs are mostly evaluated with graph data partitioned artificially. Nevertheless, the long-term development of this field still requires realistic and large-scale federated graph datasets to be made available to support experimental evaluations under settings close to practical applications. Real-world graph datasets, such as healthcare datasets, recommender systems, and KGs, can be useful starting points.

VIII. Conclusion

In this survey, we provide an overview of FedGNNs. We propose a 2-D taxonomy of FedGNNs, categorizing them based on system settings and client heterogeneity. In addition, we highlight key challenges, possible strategies, and the advantages and disadvantages associated with each category. Furthermore, we discuss commonly employed applications, accompanied by relevant public datasets, evaluation metrics, and experimental designs in the FedGNN literature. We also outline open problems and directions that would inspire further research in FedGNNs. We believe that the discussions in this survey based on our proposed FedGNN taxonomy will provide a useful roadmap for aspiring researchers and practitioners seeking to enter the field of FedGNNs and contribute to its long-term development.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Mar. 2020.

[2] 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. (ICML)*, 2017, pp. 1263–1272.

[3] Y. Ding, N. Robinson, C. Tong, Q. Zeng, and C. Guan, "LGGNet: Learning from local-global-graph representations for brain–computer interface," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–14, 2023.

[4] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 1024–1034.

[5] Y. Chen, L. Wu, and M. J. Zaki, "Toward subgraph-guided knowledge graph question generation with graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 24, 2023, doi: 10.1109/TNNLS.2023.3264519.

[6] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 974–983.

[7] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4883–4894, Nov. 2020.

[8] GDPR. (2018). *General Data Protection Regulation*. Accessed: Dec. 8, 2021. [Online]. Available: https://gdpr-info.eu/

[9] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[10] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, Jun. 2021.

[11] A. Li, L. Zhang, J. Tan, Y. Qin, J. Wang, and X.-Y. Li, "Sample-level data selection for federated learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2021, pp. 1–10.

[12] A. Li et al., "Efficient federated-learning model debugging," in *Proc. IEEE 37th Int. Conf. Data Eng. (ICDE)*, Apr. 2021, pp. 372–383.

[13] R. Goebel, H. Yu, B. Faltings, L. Fan, and Z. Xiong, *Trustworthy Federated Learning*, vol. 13448. Cham, Switzerland: Springer, 2023.

[14] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, 2021.

[15] A. Li, L. Zhang, J. Wang, F. Han, and X.-Y. Li, "Privacy-preserving efficient federated-learning model debugging," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 10, pp. 2291–2303, Oct. 2022.

[16] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, pp. 1–17, 2022.

[17] S. Scardapane, I. Spinelli, and P. D. Lorenzo, "Distributed training of graph convolutional networks," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 7, pp. 87–100, 2021.

[18] H. Zhang, T. Shen, F. Wu, M. Yin, H. Yang, and C. Wu, "Federated graph learning—A position paper," 2021, *arXiv:2105.11099*.

[19] C. He et al., "FedGraphNN: A federated learning system and benchmark for graph neural networks," 2021, *arXiv:2104.07145*.

[20] X. Fu, B. Zhang, Y. Dong, C. Chen, and J. Li, "Federated graph machine learning: A survey of concepts, techniques, and applications," *ACM SIGKDD Explorations Newslett.*, vol. 24, no. 2, pp. 32–47, Nov. 2022.

[21] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, *Federated Learning*. Cham, Switzerland: Springer, 2019.

[22] 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.

[23] Z. Wu, X. Wu, and Y. Long, "Multi-level federated graph learning and self-attention based personalized Wi-Fi indoor fingerprint localization," *IEEE Commun. Lett.*, vol. 26, no. 8, pp. 1794–1798, Aug. 2022.

[24] P. Xing, S. Lu, L. Wu, and H. Yu, "BiG-fed: Bilevel optimization enhanced graph-aided federated learning," *IEEE Trans. Big Data*, pp. 1–12, 2022.

[25] F. Chen, G. Long, Z. Wu, T. Zhou, and J. Jiang, "Personalized federated learning with graph," 2022, *arXiv:2203.00829*.

[26] B. Li, A. Swami, and S. Segarra, "Power allocation for wireless federated learning using graph neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 5243–5247.

[27] H. Lee, A. L. Bertozzi, J. Kovačević, and Y. Chi, "Privacy-preserving federated multi-task linear regression: A one-shot linear mixing approach inspired by graph regularization," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2022, pp. 5947–5951.

[28] Y. He, D. Yan, and F. Chen, "Hierarchical federated learning with local model embedding," *Eng. Appl. Artif. Intell.*, vol. 123, Aug. 2023, Art. no. 106148.

[29] C. Meng, S. Rambhatla, and Y. Liu, "Cross-node federated graph neural network for spatio-temporal data modeling," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 1202–1211.

[30] D. Caldarola, M. Mancini, F. Galasso, M. Ciccone, E. Rodola, and B. Caputo, "Cluster-driven graph federated learning over multiple domains," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 2749–2758.

[31] O. Litany, H. Maron, D. Acuna, J. Kautz, G. Chechik, and S. Fidler, "Federated learning with heterogeneous architectures using graph HyperNetworks," 2022, *arXiv:2201.08459*.

[32] A. Lalitha, O. Cihan Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," 2019, *arXiv:1901.11173*.

[33] Y. Pei et al., "Decentralized federated graph neural networks," in *Proc. Int. Workshop Federated Transf. Learn. Data Sparsity Confidentiality Conjunct (IJCAI)*, 2021.

[34] E. Rizk and A. H. Sayed, "A graph federated architecture with privacy preserving learning," in *Proc. IEEE 22nd Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2021, pp. 131–135.

[35] S. Lu, Y. Zhang, and Y. Wang, "Decentralized federated learning for electronic health records," in *Proc. 54th Annu. Conf. Inf. Sci. Syst.*, 2020, pp. 1–5.

[36] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, no. 5, pp. 808–828, May 2018.

[37] C. T. Dinh, T. T. Vu, N. H. Tran, M. N. Dao, and H. Zhang, "A new look and convergence rate of federated multitask learning with Laplacian regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–11, 2022.

[38] C. He, E. Ceyani, K. Balasubramanian, M. Annavaram, and S. Avestimehr, "SpreadGNN: Serverless multi-task federated learning for graph neural networks," 2021, *arXiv:2106.02743*.

[39] H. Wang, X. Zhao, and W. Lin, "Heterogeneous federated learning on a graph," 2022, *arXiv:2209.08737*.

[40] V. C. Gogineni, S. Werner, Y. Huang, and A. Kuh, "Decentralized graph federated multitask learning for streaming data," in *Proc. 56th Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2022, pp. 101–106.

[41] P. Veličković et al., "Graph attention networks," in *Proc. Int. Conf. Learn. Represent. (ICLR'18)*, 2018, pp. 1–11.

[42] Y. Tao, Y. Li, and Z. Wu, "SemiGraphFL: Semi-supervised graph federated learning for graph classification," in *Proc. Int. Conf. Parallel Problem Solving Nature*. Cham, Switzerland: Springer, 2022, pp. 474–487.

[43] X. Yuan et al., "FedSTN: Graph representation driven federated learning for edge computing enabled urban traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, pp. 1–11, 2022.

[44] H. Du, M. Shen, R. Sun, J. Jia, L. Zhu, and Y. Zhai, "Malicious transaction identification in digital currency via federated graph deep learning," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2022, pp. 1–6.

[45] H. Peng, Y. Zhang, H. Sun, X. Bai, Y. Li, and S. Wang, "Domain-aware federated social bot detection with multi-relational graph neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 1–8.

[46] J. Guo, S. Li, and Y. Zhang, "An information theoretic perspective for heterogeneous subgraph federated learning," in *Proc. Database Syst. Adv. Appl., 28th Int. Conf. DASFAA*, Tianjin, China. Cham, Switzerland: Springer, Apr. 2023, pp. 745–760.

[47] L. Zheng, J. Zhou, C. Chen, B. Wu, L. Wang, and B. Zhang, "ASFGNN: Automated separated-federated graph neural network," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 3, pp. 1692–1704, May 2021.

[48] T. Zhang, C. Mai, Y. Chang, C. Chen, L. Shu, and Z. Zheng, "FedEgo: Privacy-preserving personalized federated graph learning with ego-graphs," *ACM Trans. Knowl. Discovery Data*, vol. 18, no. 2, pp. 1–27, Feb. 2024.

[49] Y. Lin, C. Chen, C. Chen, and L. Wang, "Improving federated relational data modeling via basis alignment and weight penalty," 2020, *arXiv:2011.11369*.

[50] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.

[51] B. Wang, A. Li, M. Pang, H. Li, and Y. Chen, "GraphFL: A federated learning framework for semi-supervised node classification on graphs," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2022, pp. 498–507.

[52] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1126–1135.

[53] S. Wang, J. Xie, M. Lu, and N. N. Xiong, "FedGraph-KD: An effective federated graph learning scheme based on knowledge distillation," in *Proc. IEEE IEEE 9th Intl Conf. Big Data Secur. Cloud (BigDataSecurity) Intl Conf. High Perform. Smart Comput., (HPSC) IEEE Intl Conf. Intell. Data Secur. (IDS)*, May 2023, pp. 130–134.

[54] W. Huang, G. Wan, M. Ye, and B. Du, "Federated graph semantic and structural learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2023, pp. 139–143.

[55] W. Zhu, J. Luo, and A. White, "Federated learning of molecular properties with graph neural networks in a heterogeneous setting," 2021, *arXiv:2109.07258*.

[56] H. Xie, J. Ma, L. Xiong, and C. Yang, "Federated graph classification over non-IID graphs," *Proc. 35th Conf. Neural Inf. Process. Syst. (NeurIPS'21)*, 2021, pp. 18839–18852.

[57] G. Wang and Q. Yan, "Federated threat detection for smart home IoT rules," in *Proc. Int. Workshop Federated Learn. Distrib. Data Mining*, 2023.

[58] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018, *arXiv:1810.00826*.

[59] Y. Liu, D. Han, J. Zhang, H. Zhu, M. Xu, and W. Chen, "Federated multi-task graph learning," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 5, pp. 1–27, Oct. 2022.

[60] H. C. Bayram and I. Rekik, "A federated multigraph integration approach for connectional brain template learning," in *Proc. Int. Workshop Multimodal Learn. Clin. Decis. Support*, Cham, Switzerland: Springer, 2021, pp. 36–47.

[61] K. Hu, J. Wu, Y. Li, M. Lu, L. Weng, and M. Xia, "FedGCN: Federated learning-based graph convolutional networks for non-Euclidean spatial data," *Mathematics*, vol. 10, no. 6, p. 1000, Mar. 2022.

[62] J. Baek, W. Jeong, J. Jin, J. Yoon, and S. J. Hwang, "Personalized subgraph federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 1396–1415.

[63] H. Xie, L. Xiong, and C. Yang, "Federated node classification over graphs with latent link-type heterogeneity," in *Proc. ACM Web Conf.*, Apr. 2023, pp. 556–566.

[64] Y. Tan, Y. Liu, G. Long, J. Jiang, Q. Lu, and C. Zhang, "Federated learning on non-IID graphs via structural knowledge sharing," in *Proc. AAAI Conf. Artif. Intell.*, vol. 37, no. 8, 2023, pp. 9953–9961.

[65] G. Lou, Y. Liu, T. Zhang, and X. Zheng, "STFL: A temporal-spatial federated learning framework for graph neural networks," 2021, *arXiv:2111.06750*.

[66] M. Jiang, T. Jung, R. Karl, and T. Zhao, "Federated dynamic graph neural networks with secure aggregation for video-based distributed surveillance," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, pp. 1–23, Aug. 2022.

[67] Z. Gürler and I. Rekik, "Federated brain graph evolution prediction using decentralized connectivity datasets with temporally-varying acquisitions," *IEEE Trans. Med. Imag.*, vol. 42, no. 7, pp. 2022–2031, Jul. 2022.

[68] H. Zhang, K. Zeng, and S. Lin, "Federated graph neural network for fast anomaly detection in controller area networks," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1566–1579, 2023.

[69] C. Wang, B. Chen, G. Li, and H. Wang, "Automated graph neural network search under federated learning framework," *IEEE Trans. Knowl. Data Eng.*, vol. 35, pp. 1–13, 2023.

[70] M. Y. Balık, A. Rekik, and I. Rekik, "Investigating the predictive reproducibility of federated graph neural networks using medical datasets," in *Proc. Int. Workshop Predictive Intell. Med.* Cham, Switzerland: Springer, 2022, pp. 160–171.

[71] F. Chen, P. Li, T. Miyazaki, and C. Wu, "FedGraph: Federated graph learning with intelligent sampling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 8, pp. 1775–1786, Aug. 2022.

[72] C. Zhang, S. Zhang, S. Yu, and J. James, "Graph-based traffic forecasting via communication-efficient federated learning," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2022, pp. 2041–2046.

[73] Z. Liu, L. Yang, Z. Fan, H. Peng, and P. S. Yu, "Federated social recommendation with graph neural network," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, pp. 1–24, Aug. 2022.

[74] C. Zhang, W. Wang, J. J. Yu, and S. Yu, "Extracting privacy-preserving subgraphs in federated graph learning using information bottleneck," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2023, pp. 109–121.

[75] Y. Djenouri, T. P. Michalak, and J. C.-W. Lin, "Federated deep learning for smart city edge-based applications," *Future Gener. Comput. Syst.*, vol. 147, pp. 350–359, Oct. 2023.

[76] Y. Liu, H. Li, X. Qian, and M. Hao, "ESA-FedGNN: Efficient secure aggregation for federated graph neural networks," *Peer-to-Peer Netw. Appl.*, vol. 16, no. 2, pp. 1257–1269, Mar. 2023.

[77] C. Zhang, S. Zhang, J. J. Q. Yu, and S. Yu, "FASTGNN: A topological information protected federated learning approach for traffic speed forecasting," *IEEE Trans. Ind. Informat.*, vol. 17, no. 12, pp. 8464–8474, Dec. 2021.

[78] C. Chen, Z. Xu, W. Hu, Z. Zheng, and J. Zhang, "FedGL: Federated graph learning framework with global self-supervision," *Inf. Sci.*, vol. 657, Feb. 2024, Art. no. 119976.

[79] K. Zhang, C. Yang, X. Li, L. Sun, and S. M. Yiu, "Subgraph federated learning with missing neighbor generation," in *Proc. 35th Conf. Neural Inf. Process. Syst. (NeurIPS'21)*, 2021.

[80] L. Peng, N. Wang, N. Dvornek, X. Zhu, and X. Li, "FedNI: Federated graph learning with network inpainting for population-based disease prediction," *IEEE Trans. Med. Imag.*, vol. 42, no. 7, pp. 2032–2043, Jul. 2022.

[81] I. Goodfellow, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[82] Y. Qiu, C. Huang, J. Wang, Z. Huang, and J. Xiao, "A privacy-preserving subgraph-level federated graph neural network via differential privacy," in *Proc. Knowl. Sci., Eng. Manage.*, 2022, pp. 6671–6682.

[83] C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, and X. Xie, "A federated graph neural network framework for privacy-preserving personalization," *Nature Commun.*, vol. 13, no. 1, pp. 1–10, Jun. 2022.

[84] C. Wu, F. Wu, Y. Cao, Y. Huang, and X. Xie, "FedGNN: Federated graph neural network for privacy-preserving recommendation," in *Proc. FL-ICML*, 2021, pp. 165–177.

[85] B. Du and C. Wu, "Federated graph learning with periodic neighbour sampling," in *Proc. IEEE/ACM 30th Int. Symp. Quality Service (IWQoS)*, Jun. 2022, pp. 1–10.

[86] Y. Yao, W. Jin, S. Ravi, and C. Joe-Wong, "FedGCN: Convergence-communication tradeoffs in federated training of graph convolutional networks," 2022, *arXiv:2201.12433*.

[87] H. Peng, H. Li, Y. Song, V. Zheng, and J. Li, "Differentially private federated knowledge graphs embedding," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manag.*, Oct. 2021, pp. 1416–1425.

[88] J. Jordon, J. Yoon, and M. Van Der Schaar, "PATE-GAN: Generating synthetic data with differential privacy guarantees," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–21.

[89] M. Chen, W. Zhang, Z. Yuan, Y. Jia, and H. Chen, "FedE: Embedding knowledge graphs in federated setting," in *Proc. 10th Int. Joint Conf. Knowl. Graphs*, Dec. 2021, pp. 80–88.

[90] K. Zhang et al., "Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation," 2022, *arXiv:2203.09553*.

[91] K. Bonawitz et al., "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1175–1191.

[92] M. Chen, W. Zhang, Z. Yuan, Y. Jia, and H. Chen, "Federated knowledge graph completion via embedding-contrastive learning," *Knowledge-Based Syst.*, vol. 252, Sep. 2022, Art. no. 109459.

[93] G. Mei, Z. Guo, S. Liu, and L. Pan, "SGNN: A graph neural network based federated learning approach by hiding structure," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2019, pp. 2560–2568.

[94] T.-H. Cheung, W. Dai, and S. Li, "Fedsgc: Federated simple graph convolution for node classification," in *Proc. Int. Workshop Federated Transf. Learn. Data Sparsity Confidentiality Conjuncation IJCAI*, 2021.

[95] C. Chen et al., "Vertically federated graph neural network for privacy-preserving node classification," in *Proc. Int. Joint Conf. Artif. Intell. Org. (IJCAI'22)*, 2022, pp. 1959–1965.

[96] W. Li and S. Wang, "Federated meta-learning for spatial–temporal prediction," *Neural Comput. Appl.*, vol. 34, no. 13, pp. 10355–10374, Jul. 2022.

[97] X. Ni, X. Xu, L. Lyu, C. Meng, and W. Wang, "A vertical federated learning framework for graph convolutional network," 2021, *arXiv:2106.11593*.
[98] J. Chen, G. Huang, H. Zheng, S. Yu, W. Jiang, and C. Cui, "Graph-fraudster: Adversarial attacks on graph neural network-based vertical federated learning," *IEEE Trans. Computat. Social Syst.*, vol. 10, no. 2, pp. 492–506, Apr. 2023.
[99] C. Wang, B. Chen, G. Li, and H. Wang, "FL-AGCNS: Federated learning framework for automatic graph convolutional network search," 2021, *arXiv:2104.04141*.
[100] S. Chen, Q. Zhu, Z. Li, and Y. Long, "Deep neural network based on feature fusion for indoor wireless localization," in *Proc. Int. Conf. Microw. Millim. Wave Technol. (ICMMT)*, May 2018, pp. 1–3.
[101] G. Karypis and V. Kumar, "Multilevelk-way partitioning scheme for irregular graphs," *J. Parallel Distrib. Comput.*, vol. 48, no. 1, pp. 96–129, 1998.
[102] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mechanics: Theory Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.
[103] Z. Wang et al., "FederatedScope-GNN: Towards a unified, comprehensive and efficient package for federated graph learning," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining (KDD'22)*, 2022, pp. 4110–4120.
[104] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[105] H. Yuan, H. Yu, S. Gui, and S. Ji, "Explainability in graph neural networks: A taxonomic survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, pp. 5782–5799, May 2023.
[106] Q. Li et al., "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3347–3366, Apr. 2023.
[107] J. Zhang and H. Yu, "EID: Facilitating explainable AI design discussions in team-based settings," *Int. J. Crowd Sci.*, vol. 7, no. 2, pp. 47–54, Jun. 2023.
[108] Z. Chen, T. Xiao, and K. Kuang, "BA-GNN: On learning bias-aware graph neural network," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, May 2022, pp. 3012–3024.
[109] Y. Dong, S. Wang, Y. Wang, T. Derr, and J. Li, "On structural explanation of bias in graph neural networks," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2022, pp. 316–326.
[110] J. Zhang, Y. Shu, and H. Yu, "Fairness in design: A framework for facilitating ethical artificial intelligence designs," *Int. J. Crowd Sci.*, vol. 7, no. 1, pp. 32–39, Mar. 2023.

Rui Liu received the B.Eng. degree from the Harbin Institute of Technology (HIT), Harbin, China, in 2014, and the Ph.D. degree from the Singapore University of Technology and Design (SUTD), Singapore, in 2019.

She is currently a Research Fellow with the School of Computer Science and Engineering (SCSE), Nanyang Technological University (NTU), Singapore. Her research focuses on graph neural networks, federated learning, and brain–computer interfaces.

Pengwei Xing received the bachelor's degree in computer science from Henan University, Kaifeng, China, in 2016, and the master's degree in computer science from Tianjin University, Tianjin, China, in 2019. He is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering (SCSE), Nanyang Technological University (NTU), Singapore.

His research mainly focuses on federated learning and graph learning.

Zichao Deng received the B.Eng. degree from Nanyang Technological University (NTU), Singapore, in 2018, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering (SCSE).

His research focuses on federated graph learning.

Anran Li received the B.S. degree from the Anhui University of Science and Technology, Huainan, China, in 2016, and the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2021.

She is currently a Research Fellow with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. Her research interests mainly focus on data quality assessment, federated learning, and mobile computing.

Cuntai Guan (Fellow, IEEE) is currently a President's Chair Professor with the School of Computer Science and Engineering, the Director of the Artificial Intelligence Research Institute, the Director of the Centre for Brain-Computing Research, and the Co-Director of the S-Lab for Advanced Intelligence, Nanyang Technological University, Singapore. His research interests include brain–computer interfaces, machine learning, neural signal and image processing, neural and cognitive rehabilitation, and artificial intelligence.

Dr. Guan is a fellow of the American Institute for Medical and Biological Engineering (AIMBE), National Academy of Inventors (NAI), USA, and the Academy of Engineering Singapore. He was a recipient of the Annual BCI Research Award, the IES Prestigious Engineering Achievement Award, the Achiever of the Year (Research) Award, the King Salman International Award for Disability Research, and the Finalist of the President Technology Award.

Han Yu (Senior Member, IEEE) received the Ph.D. degree from the School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore, in 2014.

He held the prestigious Lee Kuan Yew Post-Doctoral Fellowship (LKY PDF) from 2015 to 2018. He is currently a Nanyang Assistant Professor (NAP) with the School of Computer Science and Engineering (SCSE), NTU. He has published over 200 research papers and book chapters in leading international conferences and journals. He is a coauthor of the book *Federated Learning*, the first monograph on the topic of federated learning. His research focuses on federated learning and algorithmic fairness.

Dr. Yu is a Distinguished Member of China Computer Federation (CCF) and a Senior Member of Association for the Advancement of Artificial Intelligence (AAAI). His research works have won multiple awards from conferences and journals.