Federated Graph Machine Learning: A Survey of Concepts, Techniques, and Applications

Xingbo Fu
University of Virginia
xf3av@virginia.edu

Binchi Zhang
University of Virginia
epb6gw@virginia.edu

Yushun Dong
University of Virginia
yd6eb@virginia.edu

Chen Chen
University of Virginia
zrh6du@virginia.edu

Jundong Li
University of Virginia
jundong@virginia.edu

ABSTRACT

Graph machine learning has gained great attention in both academia and industry recently. Most of the graph machine learning models, such as Graph Neural Networks (GNNs), are trained over massive graph data. However, in many real-world scenarios, such as hospitalization prediction in healthcare systems, the graph data is usually stored at multiple data owners and cannot be directly accessed by any other parties due to privacy concerns and regulation restrictions. Federated Graph Machine Learning (FGML) is a promising solution to tackle this challenge by training graph machine learning models in a federated manner. In this survey, we conduct a comprehensive review of the literature in FGML. Specifically, we first provide a new taxonomy to divide the existing problems in FGML into two settings, namely, *FL with structured data* and *structured FL*. Then, we review the mainstream techniques in each setting and elaborate on how they address the challenges under FGML. In addition, we summarize the real-world applications of FGML from different domains. Finally, we present several limitations in the existing studies with promising research directions in this field.

KEYWORDS

federated learning, graph mining, graph neural networks

ACM Reference Format:
Xingbo Fu, Binchi Zhang, Yushun Dong, Chen Chen, and Jundong Li. 2018. Federated Graph Machine Learning: A Survey of Concepts, Techniques, and Applications. In *The First International Workshop on Federated Learning over Graph Data (FedGraph '22), October 21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

In recent years, graphs have been widely used to represent complex data in a wide diversity of real-world domains, e.g., healthcare [69], transportation [34], and recommendation systems [10]. Numerous graph machine learning techniques provide insights into understanding rich information hidden in graphs and show expressive performance in different tasks, such as node classification [89] and link prediction [4].

Although these graph machine learning techniques have made significant progress, most of them require a massive amount of graph data centrally stored on a single machine. However, with the emphasis on data security and user privacy [67], this requirement is often infeasible in the real world. Instead, graph data is usually distributed in multiple data owners (i.e., data isolation) and we are not able to collect the graph data in different places due to privacy concerns. For instance, a third-party company aims to train a graph machine learning model for a group of financial institutions to help them detect potential financial crimes and fraud customers. Each financial institution owns its local dataset of customers, such as their demographics, as well as transaction records among them. The customers in each financial institution form a customer graph where edges represent the transaction records. Due to strict privacy policies and commercial competition, local customer data in each institution cannot be directly shared with the company or other institutions. Meanwhile, some institutions may have connections with others, which could be viewed as structural information among institutions. Generally, the major challenge for the company lies in training a graph machine learning model for financial crime detection based on the local customer graphs and structural information among institutions without directly accessing local customer data in each institution.

Federated Learning (FL) [49] is a distributed learning scheme which addresses the data isolation problem through collaborative training. It enables participants (i.e., clients) to jointly train a machine learning model without sharing their private data. Therefore, combining FL with graph machine learning becomes a promising solution to the aforementioned problem. In this paper, we term it Federated Graph Machine Learning (FGML). In general, FGML can be categorized as two settings with respect to the level of structural information. The first setting is *FL with structured data*. In FL with structured data, clients collaboratively train a graph machine learning model based on their graph data while keeping the graph data locally. The second setting is *structured FL*. In structured FL, there are structural information among the clients which forms a client-level graph. The client graph could be leveraged to design more effective federated optimization approaches.

While FGML provides a promising paradigm, the following challenges emerge and need to be addressed.

FedGraph '22, October 21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

(1) *Cross-client missing information.* A common scenario in FL with structured data is that each client owns a subgraph of the global graph and some nodes may have neighbors belonging to other clients. Due to privacy concerns, a node can only aggregate the features of its neighbors within the client but cannot access the features of those on other clients, which leads to insufficient node embeddings [8, 87].
(2) *Privacy leakage of graph structures.* In traditional FL, a client is not allowed to expose the features and labels of its data samples. In Fl with structured data, the privacy of structural information should also be considered. The structural information can be either directly exposed by sharing the adjacency matrix or indirectly exposed by transmitting node embeddings [43, 58, 71, 86].
(3) *Data heterogeneity across clients.* Unlike traditional FL where data heterogeneity comes from non-IID data samples [26, 62], graph data in FGML contains rich structural information [28, 29, 42, 88]. Meanwhile, divergent graph structures across clients can also affect performance of graph machine learning models.
(4) *Parameter utilization strategies.* In structured FL, the client graph enables a client to obtain information from its neighboring clients. The effective strategies of fully utilizing neighbor information orchestrated by a central server or in a fully decentralized manner should be well designed in structured FL [24, 33, 51].

To tackle the above challenges, a great number of algorithms have been proposed in recent years. However, to the best of our knowledge, the existing surveys mainly focus on challenges and approaches in standard FL [31, 36, 77, 93] yet only a few attempts have been made to survey specific problems and techniques in FGML [40, 85]. A position paper [85] provides a categorization of FGML but does not summarize main techniques in FGML. Another review paper [40] only covers a limited number of related papers in this topic and introduces the existing techniques very briefly.

In this survey, we introduce the concepts of two problem settings in FGML. Then we review the current techniques under each setting and introduce real-world applications in FGML. Finally, several promising future directions are presented. Our contributions in this paper can be summarized as follows.

- **Taxonomy of Techniques in FGML.** We propose a taxonomy of FGML based on different problem settings and summarize key challenges in each setting.
- **Comprehensive Technique Review.** We provide a comprehensive overview of the existing techniques in FGML. Compared with the existing reviews, we not only investigate a more extensive set of related work but also provide a more elaborate analysis of techniques instead of simply listing the steps of each method.
- **Real-World Applications.** We are the first to summarize real-world applications of FGML. We categorize the applications by their domains and introduce related works in each domain.
- **Promising Future Directions.** We point out the limitations of the existing methods and provide promising future directions in FGML.

The rest of this paper is organized as follows. Section 2 briefly introduces definitions in graph machine learning as well as concepts and challenges of two settings in FGML. We review mainstream techniques in the two settings in Section 3 and Section 4, respectively. Section 5 further explores applications of FGML in the real world. We also provide possible future directions in Section 6. Finally, Section 7 concludes this paper.

2 PROBLEM FORMULATION

In this section, we first present related definitions in graph machine learning and FL. Then we introduce the problem formulation of two different settings in FGML.

Notations. Throughout this paper, we use bold lowercase letters (e.g., $\mathbf{z}$) and bold uppercase letters (e.g., $\mathbf{A}$) to represent vectors and matrices, respectively. For any matrix, e.g., $\mathbf{A}$, we use $\mathbf{A}_i$ to denote its i-th row vector and $\mathbf{A}_{ij}$ to denote its (i, j)-th entry. The l_p norm of a vector $\mathbf{z}$ for $p \geq 1$ is denoted as $||\mathbf{z}||_p$. We use letters in calligraphy font (e.g., $\mathcal{V}$) to denote sets. $|\mathcal{V}|$ denotes the cardinality of set $\mathcal{V}$.

2.1 Graph Machine Learning

Definition 1. (Graphs) *A graph is $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set and $\mathcal{E}$ is the edge set. $v_i \in \mathcal{V}$ denotes a node and $e_{ij} = (v_i, v_j) \in \mathcal{E}$ denotes an edge between node v_i and node v_j.*

We use $\mathbf{A} \in \{0, 1\}^{n \times n}$ to represent the adjacency matrix of graph $\mathcal{G}$, where $n = |\mathcal{V}|$ is the total number of nodes. $\mathbf{A}_{ij} = 1$ implies that there exists an edge between node v_i and node v_j, otherwise $\mathbf{A}_{ij} = 0$. $\mathbf{D} \in \mathbb{R}^{n \times n}$ denotes the degree diagonal matrix where $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$. The neighborhood of node v_i is defined as $N(v_i) = \{v_j \in \mathcal{V} | (v_i, v_j) \in \mathcal{E}\}$. For graph data with node features, we use $\mathbf{X} \in \mathbb{R}^{n \times d}$ to denote the node feature matrix where d is the number of node features.

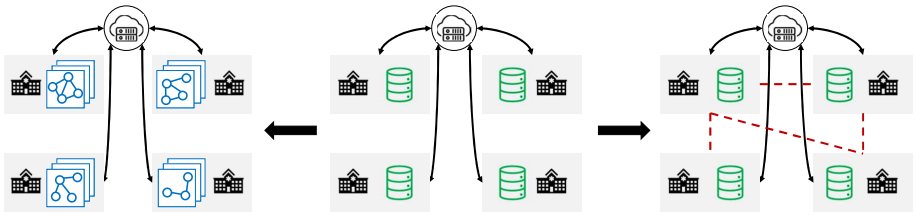
Graphs can be categorized as homogeneous graphs (containing only one type of nodes and one type of edges) and heterogeneous graphs (whose nodes belong to more than one type of nodes and/or edges) according to the number of node types and edge types. The two typical heterogeneous graphs we mention in this paper are knowledge graphs (KGs) and user-item graphs.

Definition 2. (Knowledge Graphs) *A knowledge graph is a directed heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where nodes are entities and edges are subject-property-object triple facts. Each edge of the form (head entity, relation, tail entity) (denoted as (h, r, t)) indicates a relationship r from a head entity h to a tail entity t.*

Definition 3. (User-Item Graphs) *A user-item graph is a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Users and items serve as nodes and relations between users and items serve as edges. In some scenarios, relations also exist between users and between items.*

Definition 4. (Graph Machine Learning Models) *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a graph machine learning model f_ω parameterized by ω learns the node representations $\mathbf{H} \in \mathbb{R}^{n \times d_e}$ with respect to $\mathcal{G}$ for downstream tasks, where d_e is the dimension of node embeddings*

$$\mathbf{H} = f_\omega(\mathcal{G}). \tag{1}$$



(a) FL with structured data (b) Original FL (c) Structured FL

Figure 1: The framework comparison among original FL, FL with structured data and structured FL. (a) FL with structured data: each client owns graph data, i.e., a (sub)graph or multiple graphs. (b) Original FL: data samples on clients are typically Euclidean data and no links exist among edges. (c) Structured FL: clients are connected by links and form a client graph.

For node classification tasks, we employ a softmax function to obtain the probability vector for each node based on its embedding, and then a loss function (e.g., cross entropy) is applied to measure the difference between predictions and the given node labels.

For graph classification tasks, a graph-level representation $\mathbf{h}_{\mathcal{G}}$ can be pooled from node representations through a pooling function (e.g., mean pooling) which aggregates the embeddings of all nodes in the graph into a single embedding vector.

Without loss of generality, we mainly consider Graph Neural Networks (GNNs) (e.g., GCN [32]) as graph machine learning models in this survey. In GNNs, each node v_i typically gathers the information from its neighbors $N(v_i)$ and aggregates them with its own information to update its representation $\mathbf{h}_i$. Mathematically, an L-layer GNN f_ω can be formulated as

$$\mathbf{h}_i^l = \sigma(\omega^l \cdot (\mathbf{h}_i^{l-1}, \text{Agg}(\{\mathbf{h}_j^{l-1} | v_j \in N(v_i)\}))) \tag{2}$$

for $l = 1, 2, \cdots, L$, where $\mathbf{h}_i^l$ is the representation of node v_i after the l-layer of f_ω and $\mathbf{h}_i^0 = \mathbf{X}_i$ is the raw feature of node v_i. ω^l is the learnable parameters in the l-layer of f_ω and $\text{Agg}(\cdot)$ is the aggregation operation. σ is an activation function.

2.2 Federated Learning

FGML is a type of FL which involves structural information. Before introducing the concepts of two settings in FGML, we provide the definition of FL in this subsection.

Definition 5. (Federated Learning) *In standard FL, we consider a set of M clients $C = \{c_k\}_{k=1}^M$. Each client c_k owns its private dataset $\mathcal{D}_k = \{(\boldsymbol{x}_i, y_i)_{i=1}^{N_k}\}$ sampled from its own data distribution, where $\boldsymbol{x}_i$ is the feature vector of i-th data sample and y_i is the corresponding label of the data sample. $N_k = |\mathcal{D}_k|$ is the number of data samples on client c_k and $N = \sum_{k=1}^M N_k$. Let l_k denote the loss function parameterized by ω on client c_k. The goal of FL is to optimize the overall objective function while keeping private datasets locally*

$$\min_\omega \sum_{k=1}^M \frac{N_k}{N} L_k(\omega) = \min_\omega \frac{1}{N} \sum_{k=1}^M \sum_{i=1}^{N_k} l_k(\mathbf{x}_i, y_i; \omega), \tag{3}$$

where L_k is the average loss over the local data on client c_k.

FedAvg [49] is a typical algorithm for federated optimization to obtain high model utility while preserving privacy. In FedAvg, only model parameters are transmitted between the central server and each client. Specifically, during each round t, the central server selects a subset of clients and sends them a copy of the current global model parameters ω^t for local training. Each selected client c_k updates the received copy ω_k^t by an optimizer such as stochastic gradient descent (SGD) for a variable number of iterations locally on its own dataset $\mathcal{D}_k$. Then the server collects updated model parameters ω_k^t from the selected clients and aggregates them to obtain a new global model ω^{t+1}. Finally, the server broadcasts the updated global model ω^{t+1} to clients for training in round $t+1$.

It is worthwhile to note that GNN and FL both involve an *aggregation* operation. Aggregation in the context of GNN represents the operation that a node updates its representation by aggregating information from its neighbors. Aggregation in the context of FL represents the operation that the central server collects model parameters from clients and updates the global model parameters. Following the previous survey [40], we use *GNN aggregation* and *FL aggregation* in this survey to represent two aggregation operations in GNN and FL, respectively.

2.3 Federated Graph Machine Learning

Standard FL mainly deals with tasks on Euclidean data (e.g., image classification) and equally aggregates model parameters from clients. Different from standard FL, federated graph machine learning involves structural information in federated optimization. Based on the level of structural information, FGML can be categorized as two mainstreams.

Setting 1. (FL with Structured Data) In FL with structured data, clients possess private structured datasets (i.e., graphs) and jointly train a graph machine learning model orchestrated by the central server. Fig. 1(a) illustrates the framework of FL with structured data. Formally, each client c_k owns its private local data $\mathcal{D}_k = \{\mathcal{G}_1, \mathcal{G}_2, \cdots\}$, where each $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ is a graph with its node set $\mathcal{V}_i$ and edge set $\mathcal{E}_i$. The objective of FL with structured data is that each client collaboratively trains a graph machine learning model f_ω with other clients based on its local graph dataset $\mathcal{D}_k$ while always keeping $\mathcal{D}_k$ locally. Note that each client in FL with structured data may have one single (sub)graph or multiple graphs. In general, clients train a graph machine learning model for graph-level tasks when each client c_k owns multiple graphs and N_k is the number of graphs on client c_k; on the contrary, when each client c_k owns one single graph $\mathcal{G}_k$ or a subgraph $\mathcal{G}_k$ of an entire

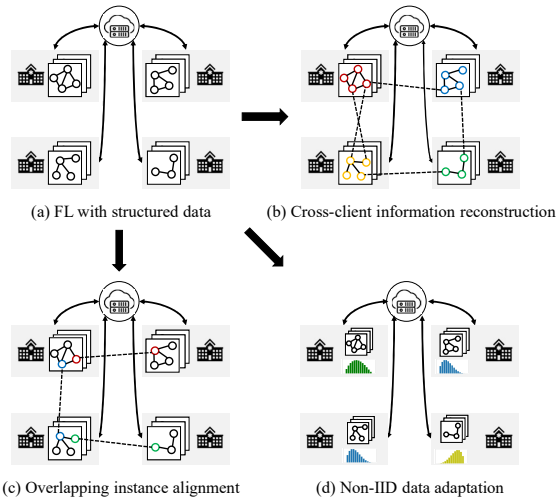


Figure 2: The taxonomy of techniques in FL with structured data. The techniques in (a) FL with structured data can be categorized as (b) cross-client information reconstruction which recovers missing links (e.g., dashed lines in (b)) between nodes from different clients, (c) overlapping instance alignment which aligns overlapping instances (e.g., nodes connected by dashed lines in (c)) among different clients, and (d) Non-IID data adaptation which tackles the non-IID characteristic of data across clients.

graph, the graph machine learning model is for node-level tasks and $N_k = |\mathcal{V}_k|$ is the number of nodes in $\mathcal{G}_k$.

Setting 2. (Structured FL) In structured FL, relations exist among clients. When we take each client as a node, all the clients in structured FL will form a graph $\mathcal{G}^C = \{\mathcal{V}^C, \mathcal{E}^C\}$ where $\mathcal{V}^C$ is the client set and $\mathcal{E}^C$ contains links between clients. Fig. 1(c) shows the framework of structured FL. Formally, given the client graph $\mathcal{G}^C = \{\mathcal{V}^C, \mathcal{E}^C\}$, each client c_k collaboratively trains a machine learning model f_ω by interacting with its neighbors $N(c_k) = \{c_s | (c_k, c_s) \in \mathcal{E}^C\}$. It is noteworthy that the datasets on clients do not have to be structured data.

In Section 3 and Section 4, we review the existing techniques in FL with structured data and structured FL and analyze how they solve the aforementioned challenges, respectively. We summarize these techniques in https://github.com/xbfu/Techniques-in-FGML.

3 FL WITH STRUCTURED DATA

The goal of clients in FL with structured data is to jointly train a graph machine learning model based on their local graph datasets while preserving privacy. In this section, we review techniques in FL with structured data for improving model utility and tackling the aforementioned challenges. Fig. 2 illustrates the taxonomy of techniques in FL with structured data.

3.1 Cross-Client Information Reconstruction

When a graph is split into multiple subgraphs and each client owns a subgraph of the original graph, each node can only perform GNN aggregation on the information from a subset of its neighbors (i.e., those within the subgraph) but cannot obtain information from those located on other clients due to the privacy issue. The missing cross-client information leads to biased node embeddings on each client and therefore degrades the performance of graph machine learning models. The objective in this case is to reconstruct the important missing cross-client information for calculating node embeddings. The existing techniques can be categorized as intermediate result transmission and missing neighbor generation. The difference lies in whether the original global graph structure is known: the studies in intermediate result transmission assume that the central server is aware of the original graph structure, while those in missing neighbor generation do not.

3.1.1 Intermediate Result Transmission. When the central server is aware of the original graph structure (including missing cross-client links), it is able to collect intermediate results (e.g., node representations) in graph machine learning models from clients and compute node embeddings according to their complete neighbor lists including cross-client neighbors.

Considering an L-layer GCN model f_ω, the operation in the l-th layer of f_ω can be written as

$$\mathbf{H}^l = \sigma\left(\mathbf{L}\mathbf{H}^{l-1}\mathbf{W}^l\right), \tag{4}$$

where $\mathbf{H}^l$ is the hidden representation after the l-th layer of f_ω and $\mathbf{H}^0 = \mathbf{X}$. $\mathbf{W}^l$ denotes the weight matrix of the l-th layer and $\mathbf{L} = (\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}$. In the federated setting, we rewrite the GCN model in a distributed manner, where the hidden matrix $\mathbf{H}^l_{(k)}$ after the l-th layer of f_ω on client c_k can be computed by

$$\begin{aligned}\mathbf{H}^l_{(k)} &= \sigma\left(\sum_{s=1}^{M}\mathbf{L}_{(ks)}\mathbf{H}^{l-1}_{(s)}\mathbf{W}^l_{(s)}\right)\\ &= \sigma\left(\mathbf{L}_{(kk)}\mathbf{H}^{l-1}_{(k)}\mathbf{W}^l_{(k)} + \sum_{s\neq k}\mathbf{L}_{(ks)}\mathbf{H}^{l-1}_{(s)}\mathbf{W}^l_{(s)}\right)\end{aligned} \tag{5}$$

for $l = 1, 2, \cdots, L$, where $\mathbf{H}^0_{(k)} = \mathbf{X}_{(k)}$ are node features on client c_k. $\mathbf{W}^l_{(k)}$ is the local parameters in the l-th layer of the local graph learning model on client c_k. $\mathbf{L}_{(ks)}$ is a block of $\mathbf{L}$ corresponding to the rows of nodes in $\mathcal{V}_k$ and columns of nodes in $\mathcal{V}_s$.

An intuitive way of obtaining information from a node's neighbors located on other clients is to transmit node embeddings directly. To avoid exposing raw node features, each client performs GNN aggregation within its local subgraph when processing the first GCN layer [8, 41]

$$\mathbf{H}^1_{(k)} = \sigma(\mathbf{L}_{(kk)}\mathbf{X}_{(k)}\mathbf{W}^1_{(k)}). \tag{6}$$

Then, each client performs GNN aggregation for a node including the embeddings of its neighbors from other clients following Eq. (5) after the first GCN layer.

However, the block of degree diagonal matrix $\mathbf{D}_{(k)} \in \mathbb{R}^{N_k \times N_k}$ containing node degree information on client c_k is unknown for other clients due to privacy concerns and must be kept locally during computation. There have been a series of corresponding solutions to this problem. For instance, PPSGCN [81] reformulates

each block $\mathbf{L}_{(ks)}$ by

$$\mathbf{L}_{(ks)} = (\mathbf{D}_{(k)} + \mathbf{I})^{-\frac{1}{2}} \cdot \tilde{\mathbf{L}}_{(ks)}, \tag{7}$$

where $\tilde{\mathbf{L}}_{(ks)} = \mathbf{A}_{(ks)}(\mathbf{D}_{(k)} + \mathbf{I})^{-\frac{1}{2}}$. Therefore, Eq. (5) can be rewritten as

$$\begin{aligned}\mathbf{H}^l_{(k)} = \sigma(&\mathbf{L}_{(kk)}\mathbf{H}^{l-1}_{(k)}\mathbf{W}^l_{(k)} \\ &+ (\mathbf{D}_{(k)} + \mathbf{I})^{-\frac{1}{2}} \sum_{s\neq k} \tilde{\mathbf{L}}_{(ks)}\mathbf{H}^{l-1}_{(s)}\mathbf{W}^l_{(s)})\end{aligned} \tag{8}$$

for $l = 1, 2, \cdots, L$, where $\tilde{\mathbf{L}}_{(ks)}\mathbf{H}^{l-1}_{(s)}\mathbf{W}^l_{(s)}$ can be calculated locally within client c_s and transmitted to the server. Consequently, we can learn node representations over clients without exchanging local graph structure information.

Although the raw data (i.e., node features and structural information) are preserved with intermediate result transmission, it requires the structural information of the original graph to compute node embeddings, which might be impractical in the real world. Furthermore, intermediate results are transmitted multiple times for each local update, which also brings extra communication costs.

3.1.2 Missing Neighbor Generation. When the original graph structure is unknown to the server, the techniques in intermediate result transmission fail since the cross-subgraph links carrying important information will never be captured by any client [87]. To tackle this issue, several approaches of missing neighbor generation have been proposed recently.

The intuition of missing neighbor generation is to design a missing neighbor generator to reconstruct the features of a node's cross-subgraph neighbors on other clients [56, 87]. Concretely, each client c_k first hides a subset of nodes and related edges in its local subgraph $\mathcal{G}_k$ following a specific strategy (e.g., Breadth-First Search) [56] to form an impaired subgraph $\bar{\mathcal{G}}_k$. Then each client trains $\mathcal{X}$ a predictor parameterized by θ_d for predicting the number $\tilde{n}_i$ of the masked neighbors of each node v_i in $\bar{\mathcal{G}}_k$ and an encoder (e.g., GCN) parameterized by θ_f for predicting the features $\tilde{\mathbf{X}}_{(i)}$ of the masked neighbors by minimizing the loss

$$L^n = \lambda^d L^d(\tilde{n}_i, n_i; \theta_d) + \lambda^f L^f(\{\tilde{\mathbf{X}}_{(i)}\}_{v_i\in\bar{\mathcal{G}}_k}, \{\mathbf{X}_{(i)}\}_{v_i\in\bar{\mathcal{G}}_k}; \theta_f), \tag{9}$$

where L^d and L^f are the loss functions for the predicted number of masked neighbors and their predicted features, respectively. $\mathbf{X}_{(i)}$ is the features of node v_i's masked neighbors. A cross-subgraph feature reconstruction term [87] is introduced to L^f, aiming to recover features of cross-subgraph missing neighbors by decreasing the distance between predicted node features and the closest node feature on other clients.

3.2 Overlapping Instance Alignment

In applications, an instance (e.g., a node in homogeneous graphs or an entity in KGs) could belong to two or more clients. Under this setting, the embeddings of an overlapping instance from different clients may come from different embedding spaces during collaborative training. To tackle this problem, the overlapping instance alignment technique is proposed [7, 55, 92]. The key idea is to learn global instance embeddings based on the local instance embeddings from clients. This technique can be applied to homogeneous graphs, KGs, and user-item graphs.

3.2.1 Homogeneous Graph-Based Alignment. The existing works about instance alignment in homogeneous graphs are mainly under vertical FL [77]. In generic vertical FL, clients have overlapping nodes but differ in the feature space. Unlike generic vertical FL, structural information in graph data is also taken into account in FGML. One common scenario is that a set of nodes are located on all clients but their features and relations are different across clients [52, 92]. The central server can collect local node embeddings from clients and align overlapping node embeddings. For instance, VFGNN [92] first computes local node embeddings $\mathbf{H}_{(k)}$ on each client c_k using a graph machine learning model. Then it combines the embeddings of each node v_i via a combination strategy

$$\mathbf{H} \leftarrow \text{COMBINE}(\{\mathbf{H}_{(k)}\}_{k=1}^M), \tag{10}$$

where COMBINE$(\cdot)$ denotes a combination operator (e.g., Concat).

Another scenario is that one client only contains structural information of graph data and other clients only contain node features [14]. In this scenario, graph machine learning models cannot be simply applied on each client since structural information and node features are located in different clients. The techniques deal with this problem by protecting structural information and raw node features simultaneously during federated optimization. For example, SGNN [50] replaces the original adjacency matrix with a structural similarity matrix $\mathbf{A}^s$ where entry $\mathbf{A}^s_{ij}$ measures the structural similarity between node v_i and node v_j. SGNN computes $\mathbf{A}^s_{ij}$ by

$$\mathbf{A}^s_{ij} = \exp(-\text{dist}(\text{OD}(N(v_i)), \text{OD}(N(v_j)))), \tag{11}$$

where OD$(\cdot)$ returns a list of ordered degree given the input node list and dist$(\cdot,\cdot)$ is a distance function (e.g., dynamic time warping (DTW) [53]). Then SGNN embeds original features on the clients which only contain features using one-hot encoding and finally computes node embeddings with the structural similarity matrix. FedSGC [14] applies Homomorphic Encryption (HE) [1] for secure transmission of the adjacency matrix and node features.

3.2.2 KG-Based Alignment. Suppose in a federated KG each client owns one KG and each KG may have overlapping entities which also exist on other clients. The key technique to improve the performance of KG embedding is to align the embeddings of overlapping entities across KGs [12, 55, 86]. More specifically, after each round t, the server collects each local embedding matrix from each client to update the global embedding matrix. Then the server distributes the global embeddings to corresponding clients for subsequent local training. As the first FL framework for KGs, FedE [12] enables the server to record all the unique entities from clients with an overall entity table. The server collects the entity embedding matrix $\mathbf{E}^t_{(k)}$ of each client c_k and aligns them by

$$\mathbf{E}^t = \left(\mathbb{1} \oslash \sum_{k=1}^M \mathbf{v}_k\right) \otimes \sum_{k=1}^M \mathbf{P}_{(k)}\mathbf{E}^t_{(k)}, \tag{12}$$

where $\mathbf{E}^t$ is the global entity embedding matrix, $\mathbb{1}$ denotes an all-one vector, $\oslash$ denotes element-wise division for vectors and $\otimes$ denotes element-wise multiplication with broadcasting. $\mathbf{P}_{(k)}$ denotes client c_k's permutation matrix that maps client c_k's entity matrix to the server's entity table. $\mathbf{v}_k$ denotes client c_k's existence vectors.

As the server maintains a complete table of entity embeddings, it can easily infer a relation embedding between two entities h and

t by calculating

$$r' = \arg\max_{r} f(h, r, t), \tag{13}$$

where $f(\cdot)$ is a score function (e.g., TransE [2]) [86]. To tackle the privacy issue, FedR [86] instead aligns relation embeddings.

Instead of aligning embeddings on the server, FKGE [55] enables entity alignment between clients. Inspired by PATE-GAN [30], FKGE involves a privacy-preserving adversarial translation (PPAT) network for adversarial learning. The PPAT network employs a generator as well as a student discriminator and multiple teacher discriminators. The generator first translates aligned entities' embeddings from $\mathcal{G}_k$ into synthesized embeddings and sents them to $\mathcal{G}_s$. The student and teacher discriminators distinguish between the synthesized embeddings and ground truth embeddings in $\mathcal{G}_s$ for each pair of KGs $(\mathcal{G}_k, \mathcal{G}_s)$ which have aligned entities $\mathcal{E}_k \cap \mathcal{E}_s$ and relations $\mathcal{R}_k \cap \mathcal{R}_s$. During the alignment, only synthesized embeddings and gradients are transmitted among clients and data privacy can be guaranteed [22].

3.2.3 User-Item Graph-Based Alignment. In a federated recommendation system, each user only has a first-order local user-item subgraph with its own item rating and its neighbors located on its device. A naive method is to align the embeddings of overlapping users and items directly. However, the server can easily infer a user's user-item links by recording the items with non-zero-gradient embeddings from this user since an item embedding gets updated on the user only when the item has the rating score from the user [43].

To tackle the privacy leakage, pseudo interacted item sampling and Local Differential Privacy (LDP) [20] techniques are two common strategies [43, 70, 71]. Before sending gradients to the central server, each user u_k first samples some items that it has not interacted with (i.e., pseudo interacted items). Then it generates embedding gradients of the sampled items (e.g., using a Gaussian distribution) and combines them with the real embedding gradients. Finally, the user applies an LDP module to modify gradients by clipping and adding zero-mean Laplacian noise to gradients

$$g'_k = \text{clip}(g_k, \delta) + \text{Laplace}(0, \lambda), \tag{14}$$

where g_k is the unified gradients of user u_k including model gradients and user/item embedding gradients, $\text{clip}(g_k, \delta)$ limits g_k with the threshold δ and λ is the strength of Laplacian noise.

3.3 Non-IID Data Adaptation

The data distribution on each client may diverge a lot both in node features and graph structures [73]. Such data heterogeneity may lead to severe model divergence in the federated setting and therefore degrade the performance of the global model. The intuition of mitigating the problem is either to train an effective global model or to train specialized models for each client. The existing techniques handling this problem can be categorized as single global model-based methods and personalized model-based methods.

3.3.1 Single Global Model-Based Methods. The goal of single global model-based methods is to train a global graph machine learning model over graph data from clients. The existing techniques tackle non-IID data across clients by designing loss functions and reweighting FL aggregations and interpolating local models.

Loss Function Designing. The intuition of loss function designing is to replace the original loss function, which is just for high model utility, with a new well-designed loss function that is also targeted at data heterogeneity. A common strategy is to add regularization terms into the local loss function. For instance, to deal with relational data (e.g., KGs) heterogeneity across clients, FedAlign [39] minimizes the average Optimal Transportation (OT) distance [66] between the basis matrices in basis decomposition [61] among clients. Mathematically, to train an L-layer graph machine learning model, the regularization term on client c_k can be rewritten as

$$L_k^r = \frac{\mu}{M} \sum_{k \neq s}^{M} \sum_{l=1}^{L} \text{OT}(\mathbf{V}_{(k)}^l, \mathbf{V}_{(s)}^l) + \lambda(||\nabla L_k(\omega)||_2 - 1)^2, \tag{15}$$

where $\mathbf{V}_{(k)}^l$ is the basis of l-th layer of the local graph machine learning model on client c_k and $\text{OT}(\cdot)$ computes OT distance. The second term is a weight penalty to make the objective function quasi-Lipschitz continuous. μ and λ are hyperparameters to adjust the contributions of each term.

In addition to regularization, another strategy in loss function designing is instance reweighting. For instance, FILT+ [94] pulls the local model closer to the global by minimizing the loss discrepancy between a local model and the global model. Specifically, FILT+ reweights instances on client c_k by putting more weights on samples with less confidence in the loss function

$$L_k^u = \sum_{i=1}^{N_k} (1 - \exp(-\Phi(\mathbf{x}_i, \omega, \omega_k)))^\gamma l_k(\hat{y}_i, y_i; \omega_k), \tag{16}$$

where $\Phi(\cdot)$ is defined as

$$\Phi(\mathbf{x}_i, \omega, \omega_k) = \phi(\mathbf{x}_i, \omega_k) + \max(\phi(\mathbf{x}_i, \omega_k) - \phi(\mathbf{x}_i, \omega), 0). \tag{17}$$

Here γ is a hyperparameter and $\phi(\mathbf{x}_i, \omega)$ indicates the uncertainty of training sample $\mathbf{x}_i$ under the model ω. Generally, if the local model ω_k on client c_k is less confident about a sample $\mathbf{x}_i$ than the global model ω, this sample will obtain a higher weight in the objective function.

Inspired by the model-agnostic meta-learning (MAML) [19], some meta learning-based methods in FGML [37, 68] rewrite the loss function on each client c_k as

$$L_k^u = \sum_{i=1}^{N_k} l_k(\omega - \alpha \nabla l_k(\omega)), \tag{18}$$

where α is a hyperparameter. Although the meta learning-based methods do not minimize the discrepancy between local models and the global model, they find an initial global model which can be easily adapted by clients after performing one or a few extra local updates.

FL Aggregation Reweighting. Apart from the loss function designing, reweighting local models during FL aggregation is also a solution to deal with non-IID data. FedGCN [25] tries to reweight local model parameters via an attention mechanism. Considering an L-layer model f_ω, FedGCN assigns adaptive weights $\{\beta_k^{t,l}\}_{l=1}^L$ to the model parameter ω_k from each client c_k in round t for FL aggregation

$$\omega^{t+1,l} = \sum_{k=1}^{M} \beta_k^{t,l} \omega_k^{t,l} \tag{19}$$

for $l = 1, 2, \cdots, L$. $\beta_k^{t,l}$ can be calculated through a softmax operation of score function $\alpha_k^{t,l}$

$$\alpha_k^{t,l} = \text{Attn}(\omega_k^{t,l}, \omega^{t,l}) = \mathbf{p}_k^l[\omega_k^{t,l}; \omega^{t,l}], \tag{20}$$

where $\text{Attn}(\cdot)$ is the attention mechanism, $[\cdot;\cdot]$ indicates a concat operation, and $\mathbf{p}_k^l$ is a trainable vector. As a result, $\{\beta_k^{t,l}\}_{l=1}^L$ can dynamically measure the closeness between each local model ω_k and the global model ω.

Model Interpolation. The model interpolation technique developed based on parameter weighted average of the global and the local models. Specifically, the model ω_k^t on client c_k is a combination of its local model ω_k^{t-1} and the global model ω^t [90]

$$\omega_k^t = \alpha_k \omega_k^{t-1} + (1 - \alpha_k)\omega^t, \tag{21}$$

where α_k is a mixing weight calculated as Jensen–Shannon divergence [38] between local and global data distributions.

3.3.2 Personalized Model-Based Methods. Unlike training a single global graph machine learning model, the goal of learning personalized models is to train personalized graph machine learning models for each client. The resulting personalized models are tailored for specific clients and thus result in good performance. Formally, the objective function for training personalized graph machine learning models can be rewritten as

$$\min_{\omega_1, \omega_2, \cdots, \omega_M} \sum_{k=1}^{M} \frac{N_k}{N} L_k(\omega_k). \tag{22}$$

One common strategy for training personalized graph machine learning models is client clustering [21, 60, 82]. The intuition of client clustering is that the clients with similar data distribution can be clustered in a group and the clients in a group share the same model parameters. The basic idea of client clustering in FGML is to dynamically assign clients to multiple clusters based on their latest gradients of graph machine learning models [73, 84]. One problem here is that the clustering result is significantly influenced by the latest gradients from clients, which are usually unstable during local training [73]. GCFL+ solves this problem by taking series of gradient norms into account for client clustering.

4 STRUCTURED FL

In the real world, a client may have connections with others, such as road paths existing among traffic sensors. These connections usually contain rich information (e.g., the similarity of data distribution) among clients. Considering these connections, the clients can form a client graph. Structured FL takes the client graph $\mathcal{G}^C = \{\mathcal{V}^C, \mathcal{E}^C\}$ into account and enables a client to obtain information from its neighbors. The key techniques in structured FL can be categorized as centralized aggregation and fully decentralized transmission. Fig. 3 illustrates the taxonomy of techniques in structured FL.

4.1 Centralized Aggregation

In structured FL with the central server, there exists structural information among clients. It is natural for the server to consider the structural information while updating parameters for each client. Generally, the server first collects parameters from clients as it does in standard FL. Then it updates parameters for each client

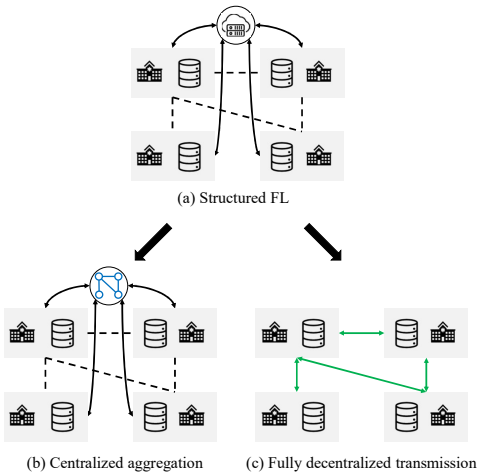


Figure 3: The taxonomy of techniques in structured FL. The techniques in (a) structured FL can be categorized as (b) centralized aggregation and (c) fully decentralized transmission.

through a graph machine learning model based on the client graph $\mathcal{G}^C$ and finally sends the updated parameters back to clients.

Transmitting Model Parameters. Transmitting local model parameters to the central server as the input of graph machine learning models is a straightforward strategy. For instance, the central server in SFL [9] collects local model parameters $\{\omega_k^t\}_{k=1}^M$ from clients in round t and employs a GCN [32] to compute a graph-based local model parameters $\{\phi_k^{t+1}\}_{k=1}^M$ by

$$\{\phi_k^{t+1}\}_{k=1}^M \leftarrow \text{GCN}(\mathbf{A}^C, \{\omega_k^t\}_{k=1}^M), \tag{23}$$

where $\mathbf{A}^C$ is the adjacency matrix of $\mathcal{G}^C$ and $\text{GCN}(\cdot)$ represents the operations in GCN. The global model parameters ω^{t+1} are calculated by a $\text{readout}(\cdot)$ operation

$$\omega^{t+1} = \text{readout}(\{\phi_k^{t+1}\}_{k=1}^M). \tag{24}$$

Each client c_k updates local mode parameters in round t by minimizing the local loss

$$\min l_k(\omega_k) + \lambda[R(\omega_k^t, \omega^t) + R(\omega_k^t, \phi_k^t)], \tag{25}$$

where $R(\cdot)$ is a regularization term.

Similarly, the server in BiG-Fed [75] collects local model parameters $\{\omega_k^t\}_{k=1}^M$ as client representatives and computes a global client embedding $\mathbf{H}$ through a graph machine learning model based on $\{\omega_k^t\}_{k=1}^M$ and the client graph. Inspired by contrastive learning [13], Big-Fed optimizes the model by

$$\begin{aligned} \min \frac{1}{n} \sum_{k=1}^{M} \sum_{c_s \in N(c_k)} & 1 - \cos(\mathbf{H}_k, \mathbf{H}_s) \\ + \frac{1}{n} \sum_{k=1}^{M} & \mathbb{E}_{c_s \sim P_k} \max(0, \cos(\mathbf{H}_k, \mathbf{H}_s)), \end{aligned} \tag{26}$$

where $\mathbf{H}_k$ is the global embedding of client c_k and P_k is a client sampling strategy. By minimizing the objective function in Eq. (26), each client will obtain a local model closer to its neighbors.

Transmitting Embeddings. Instead of gathering model parameters, another strategy for the server is to collect local embeddings from each client and compute global embeddings through graph machine learning models. For instance, CNFGNN [51] assumes that each client c_k represents a sensor with time series data $\mathbf{x}_k$. A client c_k first computes a temporal embedding $\mathbf{h}_k$ by a local encoder (e.g., GRU [15]) which models local temporal dynamics. The central server collects temporal embeddings $\{\mathbf{h}_k\}_{k=1}^M$ from clients and employs a graph machine learning model to compute the spatial embeddings $\{\mathbf{h}_k^{\mathcal{G}}\}_{k=1}^M$ of clients based on their temporal embeddings and client graph $\mathcal{G}^C$. As a result, $\mathbf{h}_k^{\mathcal{G}}$ integrates information from client c_k's neighbors and spatial dynamics. $\mathbf{h}_k^{\mathcal{G}}$ is finally sent back to client c_k as the input of a local decoder to make prediction.

4.2 Fully Decentralized Transmission

In the federated setting, one crucial bottleneck lies in high communication cost on the central server [24]. A feasible solution to tackle this issue is to train a model in a fully decentralized fashion. Since the clients in structured FL form a client graph $\mathcal{G}^C = \{\mathcal{V}^C, \mathcal{E}^C\}$, each client $c_k \in \mathcal{V}^C$ can transmit parameters with its neighbors $N(c_k)$. Specifically, when a client c_k receives model parameters $\{\omega_s^t | c_s \in N(c_k)\}$ from its neighbors $N(c_k)$ [33, 54, 57], it performs GNN aggregation to update its local model parameters ω_k^{t+1} by

$$\omega_k^{t+1} = \text{AGG}(\{\omega_s^t | c_s \in N(c_k)\}), \tag{27}$$

where $\text{AGG}(\cdot)$ is the aggregation function. An intuitive method [54, 57] following this strategy is to let each client c_k directly sum up the model parameters from its neighboring clients

$$\omega_k^{t+1} = \sum_{c_s \in N(c_k)} \mathbf{A}_{ks}^C \cdot \omega_s^t. \tag{28}$$

However, this method results in losing local information of each client since the updated model is fully determined by neighboring information. This issue can be mitigated by involving local information during aggregation [3, 17, 24, 45, 74, 79]. For example, the authors of [45, 46] directly add local gradients during aggregation. Formally, the operation can be written as

$$\omega_k^{t+1} = \sum_{c_s \in N(c_k)} \mathbf{A}_{ks}^C \cdot \omega_s^t - \alpha \nabla L_k(\omega_k^t). \tag{29}$$

Some methods [74] choose to retain local model parameters. Formally, these methods can be written as

$$\omega_k^{t+1} = \mathbf{A}_{kk}^C \cdot \omega_k^t + \sum_{c_s \in N(c_k)} \mathbf{A}_{ks}^C \cdot \omega_s^t - \alpha \nabla L_k(\omega_k^t). \tag{30}$$

5 APPLICATIONS

The number of applications of FGML is greatly increasing in various domains such as transportation, computer vision, recommendation systems, and healthcare. In this section, we elaborate some representative applications of FGML.

Transportation. Traffic prediction plays an important role in urban computing since it benefits reducing traffic congestion and improving transportation efficiency in smart cities [91]. The target of traffic prediction is to predict traffic speed or traffic flow of regions or road segments based on historical data collected by devices deployed in each region or road segment. Traffic data containing spatial-temporal information can be naturally represented as graphs and used as inputs of graph machine learning models for traffic flow prediction [44, 83]. Moreover, other applications of FGML in transportation systems, such as location representation [23, 72], routing planing [78, 80], and user mobility anomaly detection [64], are also attracting increasing attention.

Computer Vision. The existing applications of FGML in computer vision consist of image classification and object trajectory prediction [5, 6, 27]. The intuition is to construct graphs which incorporate semantic relationships among classes and objects on each client and embed them via graph machine learning models. For image classification, a graph is constructed on each client to represent classes (or domains) and the connections among them; then the clients jointly train a graph machine learning model to learn class embeddings [5, 6]. For object trajectory prediction, the idea is to construct a series of dynamic graphs from videos. Each graph represents objects and their spatial relationships in a video frame. In a federated setting (e.g., a distributed surveillance system), a dynamic GNN is transmitted for collaborative training [27].

Recommendation Systems. The downstream task in graph-based federated recommendation systems is to produce high-quality item rating for each user. A graph machine learning model learns to predict unobserved item rating to a user by leveraging the embeddings of users and items in a user's local user-item subgraph with its own item rating. Aside from transmitting model parameters, user embeddings and item embeddings are also collected by the server, which leads to information leakage of item rating becoming a primary concern in federated recommendation systems [43, 70, 71]. This privacy issue is mitigated via adding the embeddings of pseudo interacted items and noise to gradients [43, 70].

Healthcare. Sensitive medical data such as medical images and disease symptoms usually exist in isolated hospitals and medical institutes. The applications of FGML in healthcare are targeted at disease and hospitalization prediction based on medical images or health records stored in different hospitals and institutes. One key feature of healthcare applications is complex connections among medical images or health records because of patient interactions. The common strategy of modeling the connections is to construct a graph where each node represents an image or a medical record from a patient [65]. In the federated setting, the existing methods deal with cross-hospital links either by reconstructing cross-hospital links [56] or by transmitting parameters with nearby hospitals [3, 45, 46].

Other Applications. Apart from the aforementioned applications, FGML has manifold applications in other domains. A number of related papers have explored applications of FGML to various problems, such as human activity recognition [59], packet routing [47], drug discovery [48], and financial crime detection [63].

6 PROMISING FUTURE DIRECTIONS

In this section, we present some limitations in current studies and provide promising directions for future advances.

Data Heterogeneity of Graph Structures. Unlike the original FL where data distribution of features and labels are considered, the non-IID characteristic of graph structures is also a key challenge. Although a few papers analyze non-IID graph structures across

clients [73], few of them address this issue completely. Despite some popular approaches designed for mitigating the non-IID characteristic in standard FL [21, 26, 62], the approaches in FGML should take graph structures into account.

Secure Aggregation of Instance Embeddings. As mentioned in Section 3, the central server may collect instance embeddings from clients for instance alignment in FGML, especially in KGs and user-item graphs. The existing studies [43, 70] applying LDP techniques and pseudo instance sampling to alleviate privacy leakage can lead to performance degradation. Thus, designing an effective yet secure aggregation scheme in FGML is still an open problem.

Fairness in FGML. Fairness is an important topic in FL. Without accessing the sensitive information (e.g., gender and race) in different clients, FL models might show distinct bias against some groups of data [18]. Furthermore, we want the model to have similar performance in each client in some FL scenarios [16, 35]. Considering structural information, FGML provides many non-trivial challenges of fairness, e.g., how the structural information affects different fairness metrics in the federated training process. Novel fairness-aware FGML models are greatly expected.

Poisoning Attacks and Defenses in FGML. Recently, a few studies about poisoning attacks and defenses have been proposed [11, 76]. Apart from poisoning attacks on data features and model parameters in standard FL, poisoning attacks on graph structures can also affect collaborative training in FGML. Designing efficient attacks on graph structures in FGML and defending against such attacks could be a promising topic in security.

7 CONCLUSIONS

A large number of powerful graph machine learning models have achieved remarkable success in different domains. However, graph machine learning in a federated setting still faces a series of new challenges and therefore attracts massive attention from both researchers and practitioners. In this paper, we introduce the concepts of two problem settings in FGML. Then we review the current techniques under each setting in detail and introduce applications of FGML from different domains in the real world. In the end, some promising future directions are provided.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under grant IIS-2006844 and CNS-2154962, the 3 Cavaliers seed grant, and the 4-VA collaborative research grant.

REFERENCES

[1] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. 2018. A survey on homomorphic encryption schemes: Theory and implementation. *Comput. Surveys* (2018).
[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.
[3] Theodora S Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. 2018. Federated learning of predictive models from federated electronic health records. *International Journal of Medical Informatics* (2018).
[4] Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. 2021. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).
[5] Debora Caldarola, Massimiliano Mancini, Fabio Galasso, Marco Ciccone, Emanuele Rodola, and Barbara Caputo. 2021. Cluster-Driven Graph Federated Learning Over Multiple Domains. In *CVPR Workshops*.
[6] Arunava Chakravarty, Avik Kar, Ramanathan Sethuraman, and Debdoot Sheet. 2021. Federated learning for site aware chest radiograph screening. In *ISBI*.
[7] Chuan Chen, Weibo Hu, Ziyue Xu, and Zibin Zheng. 2021. FedGL: federated graph learning framework with global self-supervision. *arXiv preprint arXiv:2105.03170* (2021).
[8] Fahao Chen, Peng Li, Toshiaki Miyazaki, and Celimuge Wu. 2021. FedGraph: Federated Graph Learning With Intelligent Sampling. *IEEE Transactions on Parallel and Distributed Systems* (2021).
[9] Fengwen Chen, Guodong Long, Zonghan Wu, Tianyi Zhou, and Jing Jiang. 2022. Personalized Federated Learning With Graph. *arXiv preprint arXiv:2203.00829* (2022).
[10] Huiyuan Chen, Lan Wang, Yusan Lin, Chin-Chia Michael Yeh, Fei Wang, and Hao Yang. 2021. Structured graph convolutional networks with stochastic masks for recommender systems. In *SIGIR*.
[11] Jinyin Chen, Guohan Huang, Haibin Zheng, Shanqing Yu, Wenrong Jiang, and Chen Cui. 2022. Graph-Fraudster: Adversarial Attacks on Graph Neural Network-Based Vertical Federated Learning. *IEEE Transactions on Computational Social Systems* (2022).
[12] Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. 2021. Fede: Embedding knowledge graphs in federated setting. In *IJCKG*.
[13] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *ICML*.
[14] Tsz-Him Cheung, Weihang Dai, and Shuhan Li. 2021. FedSGC: Federated Simple Graph Convolution for Node Classification. In *IJCAI Workshops*.
[15] Kyunghyun Cho, Bart van Merrienboer, Caglar Gucehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.
[16] Sen Cui, Weishen Pan, Jian Liang, Changshui Zhang, and Fei Wang. 2021. Addressing algorithmic disparity and performance inconsistency in federated learning. In *NeurIPS*.
[17] Canh T Dinh, Tung T Vu, Nguyen H Tran, Minh N Dao, and Hongyu Zhang. 2021. A new look and convergence rate of federated multi-task learning with Laplacian regularization. *arXiv preprint arXiv:2102.07148* (2021).
[18] Yahya H Ezzeldin, Shen Yan, Chaoyang He, Emilio Ferrara, and Salman Avestimehr. 2021. Fairfed: Enabling group fairness in federated learning. In *NeurIPS Workshops*.
[19] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.
[20] Robin C Geyer, Tassilo Klein, and Moin Nabi. 2017. Differentially private federated learning: A client level perspective. In *NIPS Workshops*.
[21] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. In *NeurIPS*.
[22] Zeli Guan, Yawen Li, Zhe Xue, Yuxin Liu, Hongrui Gao, and Yingxia Shao. 2021. Federated graph neural network for cross-graph node classification. In *CCIS*.
[23] Saket Gurukar, Srinivasan Parthasarathy, Rajiv Ramnath, Catherine Calder, and Sobhan Moosavi. 2021. Locationtrails: A federated approach to learning location embeddings. In *ASONAM*.
[24] Chaoyang He, Emir Ceyani, Keshav Balasubramanian, Murali Annavaram, and Salman Avestimehr. 2022. Spreadgnn: Serverless multi-task federated learning for graph neural networks. In *AAAI*.
[25] Kai Hu, Jiasheng Wu, Yaogen Li, Meixia Lu, Liguo Weng, and Min Xia. 2022. FedGCN: Federated learning-based graph convolutional networks for non-Euclidean spatial data. *Mathematics* (2022).
[26] Yutao Huang, Lingyang Chu, Zirui Zhou, Lanjun Wang, Jiangchuan Liu, Jian Pei, and Yong Zhang. 2021. Personalized cross-silo federated learning on non-IID data.. In *AAAI*.
[27] Meng Jiang, Taeho Jung, Ryan Karl, and Tong Zhao. 2022. Federated dynamic graph neural networks with secure aggregation for video-based distributed surveillance. *ACM Transactions on Intelligent Systems and Technology* (2022).
[28] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *KDD*.
[29] Daniel Johnson, Hugo Larochelle, and Daniel Tarlow. 2020. Learning graph structure with a finite-state automaton layer. In *NeurIPS*.
[30] James Jordon, Jinsung Yoon, and Mihaela Van Der Schaar. 2018. PATE-GAN: Generating synthetic data with differential privacy guarantees. In *ICLR*.
[31] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* (2021).
[32] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[33] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. 2019. Peer-to-peer federated learning on graphs. *arXiv preprint arXiv:1901.11173* (2019).
[34] Mengzhang Li and Zhanxing Zhu. 2021. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In *AAAI*.

[35] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and robust federated learning through personalization. In *ICML*.

[36] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* (2020).

[37] Wenzhu Li and Shuang Wang. 2022. Federated meta-learning for spatial-temporal prediction. *Neural Computing and Applications* (2022).

[38] J Lin and SKM Wong. 1990. A new directed divergence measure and its characterization. *International Journal of General System* (1990).

[39] Yilun Lin, Chaochao Chen, Cen Chen, and Li Wang. 2020. Improving federated relational data modeling via basis alignment and weight penalty. *arXiv preprint arXiv:2011.11369* (2020).

[40] Rui Liu and Han Yu. 2022. Federated graph neural networks: Overview, techniques and challenges. *arXiv preprint arXiv:2202.07256* (2022).

[41] Tao Liu, Peng Li, and Yu Gu. 2021. Glint: Decentralized federated graph learning with traffic throttling and flow scheduling. In *IWQOS*.

[42] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. 2022. Towards unsupervised deep graph structure learning. In *WWW*.

[43] Zhiwei Liu, Liangwei Yang, Ziwei Fan, Hao Peng, and Philip S Yu. 2021. Federated social recommendation with graph neural network. *ACM Transactions on Intelligent Systems and Technology* (2021).

[44] Savita Lonare and Ravi Bhramaramba. 2021. Federated approach for privacy-preserving traffic prediction using graph convolutional network. *Journal of Shanghai Jiaotong University (Science)* (2021).

[45] Songtao Lu, Yawen Zhang, and Yunlong Wang. 2020. Decentralized federated learning for electronic health records. In *CISS*.

[46] Songtao Lu, Yawen Zhang, Yunlong Wang, and Christina Mack. 2019. Learn electronic health records by fully decentralized federated learning. In *NeurIPS Workshops*.

[47] Xuan Mai, Quanzhi Fu, and Yi Chen. 2021. Packet routing with graph attention multi-agent reinforcement learning. In *GLOBECOM*.

[48] Daniel Manu, Yi Sheng, Junhuan Yang, Jieren Deng, Tong Geng, Ang Li, Caiwen Ding, Weiwen Jiang, and Lei Yang. 2021. FL-DISCO: Federated generative adversarial network for graph-based molecule drug discovery. In *ICCAD*.

[49] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*.

[50] Guangxu Mei, Ziyu Guo, Shijun Liu, and Li Pan. 2019. Sgnn: A graph neural network based federated learning approach by hiding structure. In *BigData*.

[51] Chuizheng Meng, Sirisha Rambhatla, and Yan Liu. 2021. Cross-node federated graph neural network for spatio-temporal data modeling. In *KDD*.

[52] Xiang Ni, Xiaolong Xu, Lingjuan Lyu, Changhua Meng, and Weiqiang Wang. 2021. A vertical federated learning framework for graph convolutional network. *arXiv preprint arXiv:2106.11593* (2021).

[53] Niels Lundtorp Olsen, Bo Markussen, and Lars Lau Raket. 2018. Simultaneous inference for misaligned multivariate functional data. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* (2018).

[54] Yang Pei, Renxin Mao, Yang Liu, Chaoran Chen, Shifeng Xu, Feng Qiang, and Blue Elephant Tech. 2021. Decentralized federated graph neural networks. In *IJCAI Workshops*.

[55] Hao Peng, Haoran Li, Yangqiu Song, Vincent Zheng, and Jianxin Li. 2021. Differentially private federated knowledge graphs embedding. In *CIKM*.

[56] Liang Peng, Nan Wang, Nicha Dvornek, Xiaofeng Zhu, and Xiaoxiao Li. 2022. FedNI: Federated graph learning with network inpainting for population-based disease prediction. *IEEE Transactions on Medical Imaging* (2022).

[57] Elsa Rizk and Ali H Sayed. 2021. A graph federated architecture with privacy preserving learning. In *SPAWC*.

[58] Sisong Ru, Bingbing Zhang, Yixin Jie, Chi Zhang, Lingbo Wei, and Chengjie Gu. 2021. Graph neural networks for privacy-preserving recommendation with Secure Hardware. In *NaNA*.

[59] Abhishek Sarkar, Tanmay Sen, and Ashis Kumar Roy. 2021. GraFeHTy: Graph neural network using federated learning for human activity recognition. In *ICMLA*.

[60] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. 2020. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[61] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*.

[62] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. 2021. Personalized federated learning using hypernetworks. In *ICML*.

[63] Toyotaro Suzumura, Yi Zhou, Natahalie Baracaldo, Guangnan Ye, Keith Houck, Ryo Kawahara, Ali Anwar, Lucia Larise Stavarache, Yuji Watanabe, Pablo Loyola, et al. 2019. Towards federated graph learning for collaborative financial crimes detection. In *NeurIPS Workshops*.

[64] Yingjie Tang, Ruijuan Jia, Xiaoming Zhou, Zhao Li, Hao Jin, and Chenglin Zhao. 2021. Federated learning of user mobility anomaly based on graph attention networks. In *ICCC*.

[65] Anshul Thakur, Pulkit Sharma, and David A Clifton. 2021. Dynamic neural graphs based federated reptile for semi-supervised multi-tasking in healthcare applications. *IEEE Journal of Biomedical and Health Informatics* (2021).

[66] Cédric Villani. 2009. *Optimal transport: old and new*.

[67] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* (2017).

[68] Binghui Wang, Ang Li, Hai Li, and Yiran Chen. 2020. Graphfl: A federated learning framework for semi-supervised node classification on graphs. *arXiv preprint arXiv:2012.04187* (2020).

[69] Zifeng Wang, Rui Wen, Xi Chen, Shilei Cao, Shao-Lun Huang, Buyue Qian, and Yefeng Zheng. 2021. Online disease diagnosis with inductive heterogeneous graph convolutional networks. In *WWW*.

[70] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. In *ICML Workshops*.

[71] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Tao Qi, Yongfeng Huang, and Xing Xie. 2022. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications* (2022).

[72] Zheshun Wu, Xiaoping Wu, and Yunliang Long. 2022. Multi-level federated graph learning and self-attention based personalized Wi-Fi indoor fingerprint localization. *IEEE Communications Letters* (2022).

[73] Han Xie, Jing Ma, Li Xiong, and Carl Yang. 2021. Federated graph classification over non-iid graphs. In *NeurIPS*.

[74] Hong Xing, Osvaldo Simeone, and Suzhi Bi. 2020. Decentralized federated learning via SGD over wireless D2D networks. In *SPAWC*.

[75] Pengwei Xing, Songtao Lu, Lingfei Wu, and Han Yu. 2021. BiG-Fed: Bilevel Optimization enhanced graph-aided federated learning. In *ICML Workshops*.

[76] Jing Xu, Rui Wang, Kaitai Liang, and Stjepan Picek. 2022. More is better (mostly): On the backdoor attacks in federated graph neural networks. *arXiv preprint arXiv:2202.03195* (2022).

[77] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology* (2019).

[78] Minghao Ye, Junjie Zhang, Zehua Guo, and H Jonathan Chao. 2021. Federated traffic engineering with supervised learning in multi-region networks. In *ICNP*.

[79] Shahryar Zehtabi, Seyyedali Hosseinalipour, and Christopher G Brinton. 2022. Decentralized event-triggered federated learning with heterogeneous communication thresholds. *arXiv preprint arXiv:2204.03726* (2022).

[80] Tengchan Zeng, Jianlin Guo, Kyeong Jin Kim, Kieran Parsons, Philip Orlik, Stefano Di Cairano, and Walid Saad. 2021. Multi-task federated learning for traffic prediction and its application to route planning. In *IV*.

[81] Binchi Zhang, Minnan Luo, Shangbin Feng, Ziqi Liu, Jun Zhou, and Qinghua Zheng. 2021. PPSGCN: A privacy-preserving subgraph sampling based distributed GCN training method. *arXiv preprint arXiv:2110.12906* (2021).

[82] Chenhan Zhang, Lei Cui, Shui Yu, and JQ James. 2021. A communication-efficient federated learning scheme for IoT-based traffic forecasting. *IEEE Internet of Things Journal* (2021).

[83] Chenhan Zhang, Shuyu Zhang, JQ James, and Shui Yu. 2021. FASTGNN: A topological information protected federated learning approach for traffic speed forecasting. *IEEE Transactions on Industrial Informatics* (2021).

[84] Chenhan Zhang, Shiyao Zhang, Shui Yu, and JQ James. 2022. Graph-based traffic forecasting via communication-efficient federated learning. In *WCNC*.

[85] Huanding Zhang, Tao Shen, Fei Wu, Mingyang Yin, Hongxia Yang, and Chao Wu. 2021. Federated graph learning–A position paper. *arXiv preprint arXiv:2105.11099* (2021).

[86] Kai Zhang, Yu Wang, Hongyi Wang, Lifu Huang, Carl Yang, and Lichao Sun. 2022. Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation. In *ACL Workshops*.

[87] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. 2021. Subgraph federated learning with missing neighbor generation. In *NeurIPS*.

[88] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. 2021. Data augmentation for graph neural networks. In *AAAI*.

[89] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. 2021. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *WSDM*.

[90] Longfei Zheng, Jun Zhou, Chaochao Chen, Bingzhe Wu, Li Wang, and Benyu Zhang. 2021. ASFGNN: Automated separated-federated graph neural network. *Peer-to-Peer Networking and Applications* (2021).

[91] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: Concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology* (2014).

[92] Jun Zhou, Chaochao Chen, Longfei Zheng, Huiwen Wu, Jia Wu, Xiaolin Zheng, Bingzhe Wu, Ziqi Liu, and Li Wang. 2022. Vertically federated graph neural network for privacy-preserving node classification. In *IJCAI*.

[93] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. 2021. Federated learning on non-IID data: A survey. *Neurocomputing* (2021).

[94] Wei Zhu, Andrew White, and Jiebo Luo. 2021. Federated learning of molecular properties in a heterogeneous setting. *arXiv preprint arXiv:2109.07258* (2021).