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

Xingbo Fu
University of Virginias
xf3av@virginia.edu

Binchi Zhang
University of Virginias
epb6gw@virginia.edu

Yushun Dong
University of Virginias
yd6eb@virginia.edu

Chen Chen
University of Virginias
zrh6du@virginia.edu

Jundong Li
University of Virginias
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:

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.
(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 in detail and introduces the existing techniques very briefly. 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., \(x\)) and bold uppercase letters (e.g., \(A\)) to represent vectors and matrices, respectively. For any matrix, e.g., \(A\), we use \(A_i\) to denote its \(i\)-th row vector and \(A_{ij}\) to denote its \((i, j)\)-th entry. The \(l_p\) norm of a vector \(x\) for \(p \geq 1\) is denoted as \(\|x\|_p\). 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 \(G = (\mathcal{V}, \mathcal{E})\), 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 \(A \in \{0, 1\}^{n \times n}\) to represent the adjacency matrix of graph \(G\), where \(n = |\mathcal{V}|\) is the total number of nodes. \(A_{ij} = 1\) implies that there exists an edge between node \(v_i\) and node \(v_j\), otherwise \(A_{ij} = 0\). \(D \in \mathbb{R}^{n \times n}\) denotes the degree diagonal matrix where \(D_{ii} = \sum_j A_{ij}\). 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 \(X \in \mathbb{R}^{n \times d}\) 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 \(G = (\mathcal{V}, \mathcal{E})\) 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 \(G = (\mathcal{V}, \mathcal{E})\). 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 \(G = (\mathcal{V}, \mathcal{E})\), a graph machine learning model \(f_G\) parameterized by \(\omega\) learns the node representations \(H \in \mathbb{R}^{n \times d_e}\) with respect to \(G\) for downstream tasks, where \(d_e\) is the dimension of node embeddings

\[
H = f_G(G). \tag{1}
\]
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 \( h_G \) 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 \( f_\omega \) can be formulated as
\[
\mathbf{h}_i^l = \sigma(\omega^l \cdot \mathbf{h}_i^{l-1}, \text{Agg}((\mathbf{h}_j^{l-1}|v_j \in N(v_i))))
\]
for \( l = 1, 2, \cdots, L \), where \( \mathbf{h}_i^l \) is the representation of node \( v_i \) after the \( l \)-layer of \( f_\omega \) and \( \mathbf{h}_i^0 = \mathbf{x}_i \) is the raw feature of node \( v_i \), \( \omega^l \) is the learnable parameters in the \( l \)-layer of \( f_\omega \) and \( \text{Agg}(\cdot) \) is the aggregation operation. \( \sigma \) 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 \( D_k = \{ (x_i, y_i) \}_{i=1}^{N_k} \) sampled from its own data distribution, where \( x_i \) is the feature vector of \( i \)-th data sample and \( y_i \) is the corresponding label of the data sample. \( N_k = |D_k| \) is the number of data samples on client \( c_k \) and \( N = \sum_{k=1}^{M} N_k \). Let \( l_k(\cdot) \) denote the loss function parameterized by \( \omega \) on client \( c_k \). The goal of FL is to optimize the overall objective function while keeping private datasets locally
\[
\min_{\omega} \frac{1}{N} \sum_{k=1}^{M} \frac{N_k}{N} l_k(\omega) = \min_{\omega} \frac{1}{N} \sum_{k=1}^{M} \sum_{i=1}^{N_k} l_k(x_i, y_i; \omega),
\]
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 \( \omega^t \) for local training. Each selected client \( c_k \) updates the received copy \( \omega_k^t \) by an optimizer such as stochastic gradient descent (SGD) for a variable number of iterations locally on its own dataset \( D_k \). Then the server collects updated model parameters \( \omega_k^{t+1} \) from the selected clients and aggregates them to obtain a new global model \( \omega^{t+1} \). Finally, the server broadcasts the updated global model 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 \textit{GNN aggregation} and \textit{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 \( D_k = \{G_1, G_2, \cdots\} \), where each \( G_i = (V_i, E_i) \) is a graph with its node set \( V \) and edge set \( E \). The objective of FL with structured data is that each client collaboratively trains a graph machine learning model \( f_\omega \) with other clients based on its local graph dataset \( D_k \) while always keeping \( D_k \) 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 \( G_k \) or a subgraph \( G_k \) of an entire...
3 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 $f_{\omega}$, the operation in the $l$-th layer of $f_{\omega}$ can be written as

$$H^l = \sigma(W H^{l-1} W^l),$$  \hspace{1cm} (4)$$

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

$$H^l_{(k)} = \sigma \left( \sum_{s=1}^{M} L_{(k)s} H^{l-1}_{(s)} W^l_{(s)} \right)$$

$$= \sigma \left( L_{(kk)} H^{l-1}_{(k)} W^l_{(k)} + \sum_{s \neq k} L_{(ks)} H^{l-1}_{(s)} W^l_{(s)} \right)$$  \hspace{1cm} (5)$$

for $l = 1, 2, \cdots, L$, where $H^0_{(k)} = X_{(k)}$ are node features on client $c_k$, $W^l_{(k)}$ is the local parameters in the $l$-th layer of the local graph learning model on client $c_k$. $L_{(k)s}$ is a block of $L$ 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 \[8, 41\]

$$H^1_{(k)} = \sigma (L_{(kk)} X_{(k)} W^1_{(k)}).$$  \hspace{1cm} (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 $D_{(k)} \in \mathbb{R}^{N_k \times N_k}$ 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 \( L_{(k)} \) by
\[
L_{(k)} = (D_{(k)} + I)^{-\frac{1}{2}} \cdot L_{(k)},
\]
where \( L_{(k)} = A_{(k)}(D_{(k)} + I)^{-\frac{1}{2}} \). Therefore, Eq. (5) can be rewritten as
\[
H_{(k)}^{l} = \sigma(L_{(kk)}H_{(k)}^{l-1}W_{(k)}^{l}) + (D_{(k)} + I)^{-\frac{1}{2}} \sum_{k \neq k} L_{(k)}H_{(k)}^{l-1}W_{(k)}^{l}
\]
for \( l = 1, 2, \ldots, L \), where \( L_{(k)}H_{(k)}^{l-1}W_{(k)}^{l} \) can be calculated locally within client \( c_k \) 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.

### 3.2 Overlapping Instance Alignment

#### 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, VF2GNN [92] first computes local node embeddings \( H_{(k)} \) on each client \( c_k \) using a graph machine learning model. Then it combines the embeddings of each node \( v_l \) via a combination strategy
\[
H \leftarrow \text{COMBINE}((H_{(k)})_{k=1}^{M}),
\]
where COMBINE(·) 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 \( A^s \) where entry \( A^s_{ij} \) measures the structural similarity between node \( v_i \) and node \( v_j \). SGNN computes \( A^s_{ij} \) by
\[
A^s_{ij} = \exp(-\text{dist}(\text{OD}(N(v_i)), \text{OD}(N(v_j)))),
\]
where \( \text{OD}(·) \) returns a list of ordered degree given the input node list and \( \text{dist}(·, ·) \) 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.


#### 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 \( E_{(k)} \) of each client \( c_k \) and aligns them by
\[
E' = \left( \mathbb{1} \otimes \sum_{k=1}^{M} v_k \right) \otimes \sum_{k=1}^{M} P(k)E_{(k)}^t
\]
where \( E' \) is the global entity embedding matrix, \( \mathbb{1} \) denotes an all-one vector, \( \otimes \) denotes element-wise division for vectors and \( \otimes \) denotes element-wise multiplication with broadcasting. \( P(k) \) denotes client \( c_k \) ’s permutation matrix that maps client \( c_k \) ’s entity matrix to the server’s entity table. \( v_k \) 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
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 entity’s embeddings from $G_k$ into synthesized embeddings and sends them to $G_t$. The student and teacher discriminators distinguish between the synthesized embeddings and ground truth embeddings in $G_t$ for each pair of KGs ($G_k, G_t$) which have aligned entities $E_k \cap E_s$ and relations $R_k \cap R_s$. During the alignment, only synthesized embeddings and gradients are transmitted among clients and data privacy can be guaranteed [22].

### 3.3.3 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 server. 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 server.

To tackle the privacy leakage, pseudo interacted item sampling [43, 70, 71]. Before sending gradients to the central server, each user only samples a subset of 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

$$g^k_t = \text{clip}(g^k_t, \delta) + \text{Laplace}(0, \lambda),$$

where $g^k_t$ is the unified gradients of user $u^k$ including model gradients and user/entity embedding gradients, clip($g^k_t, \delta$) limits $g^k$ with the threshold $\delta$ and $\lambda$ 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

$$L^k = \frac{\mu}{M} \sum_{k=1}^{M} \sum_{l=1}^{L} \text{OT}(V^l(k), \Phi(k)) + \lambda (||\nabla L_k(\omega)|| - 1)^2,$$

where $V^l(k)$ 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. $\mu$ and $\lambda$ 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

$$L^k = \frac{N_k}{\sum_{l=1}^{L} N_k} (1 - \exp(-\Phi(x_i, \omega, \omega^k)))^2 L_k(\hat{y}_i, y_i; \omega^k),$$

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

$$\Phi(x_i, \omega, \omega^k) = \phi(x_i, \omega^k) + \max(\phi(x_i, \omega^k) - \phi(x_i, \omega), 0).$$

Here $\gamma$ is a hyperparameter and $\phi(x_i, \omega)$ indicates the uncertainty of training sample $x_i$ under the model $\omega$. Generally, if the local model $\omega^k$ on client $c_k$ is less confident about a sample $x_i$ than the global model $\omega$, 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

$$L^k = \sum_{l=1}^{N_k} I_k(\omega - a\nabla L_k(\omega)),$$

where $a$ 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 $f_{o,i}$, FedGCN assigns adaptive weights $\beta^L_k l=1$ to the model parameter $\omega^k$ from each client $c_k$ in round $t$ for FL aggregation

$$\omega^{t+1} = \sum_{k=1}^{K} \beta^L_k \omega^k, t^L$$
for \(l = 1, 2, \cdots, L\), \(\beta_{k}^{L} \) can be calculated through a softmax operation of score function \(\alpha_{k}^{L} \)
\[
\alpha_{k}^{L} = \text{Attn}(\omega_{k}^{L}, \omega^{L}) = p_{k}^{L} [\omega_{k}^{L}, \omega^{L}],
\]
where \(\text{Attn}()\) is the attention mechanism, \([::]\) indicates a concat operation, and \(p_{k}^{L}\) is a trainable vector. As a result, \([p_{k}^{L}]_{L=1}^{L}\) can dynamically measure the closeness between each local model \(\omega_{k}\) and the global model \(\omega\).

**Model Interpolation.** The model interpolation technique developed based on parameter weighted average of the global and the local models. Specifically, the model \(\omega_{k}^{t}\) on client \(c_{k}\) is a combination of its local model \(\omega_{k}^{t-1}\) and the global model \(\omega^{t}\) [90]
\[
\omega_{k}^{t} = \alpha_{k}^{t} \omega_{k}^{t-1} + (1 - \alpha_{k}) \omega^{t},
\]
where \(\alpha_{k}\) 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
\[
\min_{\omega_{k}, \omega_{k}, \cdots, \omega_{M}} \sum_{k=1}^{M} \frac{N_{k}}{N} \mathcal{L}_{k}(\omega_{k}).
\]

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 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 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 \(\omega_{k}^{M}\) from clients in round \(t\) and employs a GCN [32] to compute a graph-based local model parameters \(\phi_{k}^{M}\) by
\[
\phi_{k}^{M} \leftarrow \text{GCN}(\mathcal{A}^{C}, \omega_{k}^{M}).
\]

Each client \(c_{k}\) updates local mode parameters in round \(t\) by minimizing the local loss
\[
\min I_{k}(\omega_{k}) + \lambda \left[ R(\omega_{k}^{L}, \omega^{L}) + R(\omega_{k}^{L}, \phi_{k}^{M}) \right],
\]
where \(R(\cdot)\) is a regularization term.

Similarly, the server in Big-Fed [75] collects local model parameters \(\omega_{k}^{M}\) as client representatives and computes a global client embedding \(\mathbf{H}\) through a graph machine learning model based on \(\omega_{k}^{M}\) and the client graph. Inspired by contrastive learning [13], Big-Fed optimizes the model by
\[
\min_{\mathbf{H}} \frac{1}{m} \sum_{k=1}^{M} \sum_{c_{k} \in N(c_{k})} 1 - \cos(\mathbf{H}_{k}, \mathbf{H}_{c}) + \frac{1}{n} \sum_{k=1}^{M} \mathbb{E}_{c_{k}, p_{k}} \max(0, \cos(\mathbf{H}_{k}, \mathbf{H}_{c})),
\]
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.

![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.](image-url)
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 $\{\mathbf{h}_k^G\}_{k=1}^M$ of clients based on their temporal embeddings and client graph $G^C$. As a result, $\mathbf{h}_k^G$ integrates information from client $c_k$’s neighbors and spatial dynamics. $\mathbf{h}_k^G$ 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 $G^C = (\mathcal{V}^C, E^C)$, 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 $\{\omega_i^0|c_i \in N(c_k)\}$ from its neighbors $N(c_k)$ [33, 54, 57], it performs GNN aggregation to update its local model parameters $\omega_k^{t+1}$ by

$$\omega_k^{t+1} = \text{AGG}(\{\omega_i^0|c_i \in N(c_k)\}),$$

where AGG(·) 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

$$\omega_k^{t+1} = \sum_{c_i \in N(c_k)} A^C_{k,i} \cdot \omega_i^t.$$

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

$$\omega_k^{t+1} = \sum_{c_i \in N(c_k)} A^C_{k,i} \cdot \omega_i^t - \alpha \nabla L_k(\omega_i^t).$$

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

$$\omega_k^{t+1} = A^C_{k,k} \cdot \omega_k^t + \sum_{c_i \in N(c_k)} A^C_{k,i} \cdot \omega_i^t - \alpha \nabla L_k(\omega_i^t).$$

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 planning [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
A large number of powerful graph machine learning models have been proposed 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 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


[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 CoRR.


