# Federated Node Classification over Graphs with Latent Link-type Heterogeneity

Han Xie
Emory University
Atlanta, GA, United States
han.xie@emory.edu

Li Xiong
Emory University
Atlanta, GA, United States
lxiong@emory.edu

Carl Yang
Emory University
Atlanta, GA, United States
j.carlyang@emory.edu

## ABSTRACT

Federated learning (FL) aims to train powerful and generalized global models without putting distributed data together, which has been shown effective in various domains of machine learning. The non-IIDness of data across local clients has been a major challenge for FL. In graphs, one specifically important perspective of non-IIDness is manifested in the link-type heterogeneity underlying homogeneous graphs– the seemingly uniform links captured in most real-world networks can carry different levels of homophily or semantics of relations, while the exact sets and distributions of such latent link-types can further differ across local clients. Through our preliminary data analysis, we are motivated to design a new graph FL framework that can simultaneously discover latent link-types and model message-passing w.r.t. the discovered link-types through the collaboration of distributed local clients. Specifically, we design a multi-channel GCN model (mGCN) to differentiate the message-passing through different types of links, and devise a clustered GCN model (cGCN) by jointly training a clustering module together with mGCN through an EM-based algorithm. FL algorithms are further developed for the training of both mGCN and cGCN across clients, which effectively maintain the consistency of cluster assignments and link-type-aware model parameters without heavy communication overhead. For experiments, we synthesize multiple realistic datasets of graphs with latent heterogeneous link-types from real-world data, and partition them with different levels of link-type heterogeneity. Comprehensive experimental results and in-depth analysis have demonstrated both superior performance and rational behaviors of our proposed techniques.

## KEYWORDS

federated learning, graph mining, link-type heterogeneity, graph neural networks, clustering

**ACM Reference Format:**
Han Xie, Li Xiong, and Carl Yang. 2018. Federated Node Classification over Graphs with Latent Link-type Heterogeneity. 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


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

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

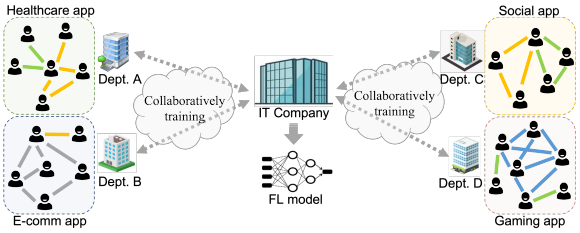



**Figure 1: Toy example of an IT company. Links in different colors represent different latent relations.** Green: friendship, yellow: kinship, gray: friends with similar purchasing preferences, blue: friends in the game.

## 1 INTRODUCTION

Federated Learning (FL) is a distributed learning paradigm in which multiple clients collaboratively train a global model without sharing their local data, in consideration of data privacy, commercial competition, resource limitation, and other regulations [30, 44]. FL has been applied to many areas including natural language processing (e.g., automatic text completion [10]) and computer vision (e.g., face recognition [9], medical images [28]). Recently, studies have also been conducted on applying FL to graph representation learning, such as FL on knowledge graphs [6], molecule classification [11], recommender systems [36], and social networks [16]. Different from applying FL on text or image data, graphs do not reside on the Euclidean space which bring a series of unique challenges for FL.

One specific challenge in FL over graph data is the data heterogeneity regarding graph links. Specifically, this can be brought by the types of links in graphs, which can be different and non-independent-identically-distributed (non-IID) across clients. Such link-types in a graph can be explicit or implicit. Explicit link-types are usually represented by a multi-view graph [3] or heterogeneous graph [29]. However, in often cases the real-world homogeneous graphs with a single explicit type of links can have different implicit link-types (relations). For example, users in a social network can be linked due to relations such as classmates, colleagues, friends, and families [39], but these relations are either unknown or private as impossible for the platforms to collect. In the FL setting, such latent link-types can vary a lot across local data owners, which makes the learning of a global unbiased graph model difficult.

Consider the scenario of FL with link-type heterogeneity based on the example of an IT company (see Figure 1). The data across different departments within the company can be biased due to the diverse functionalities of the departments. For example, departments developing social or healthcare applications are more likely to collect data where the major link-types are friendship or kinship,

which may not appear in the data collected by departments developing e-commerce or gaming applications whose major link-types are friends with similar interests. When the company develops a new service starting with a small number of users and connections or conducts a new analysis for which the most relevant link-types are unknown, it would be beneficial to collaboratively learn a graph model across all departments (clients).

In the centralized learning setting, several prior works have been done to study different link types in a single global graph. Among them, most studies focus on differentiating homophile and heterophile links [5, 7, 12] and inferring the different latent semantics of seemingly homogeneous links [31, 32, 39]. Works in the first category usually propose well-designed frameworks specifically for heterophile networks [43], while some works [7, 12] study the varying extent of homophily in graphs. Works in the second category incorporate auxiliary information such as user attributes and information flows for profiling the relations in graphs [39]. However, as we have illustrated before, data heterogeneity regarding link-types is a unique yet important challenge in FL over graph data, which has not been studied yet. In this work, we consider this unique challenge, which generalizes both the extent of homophily and semantics of relations on different links in the FL setting.

To further motivate the necessity of studying link-type heterogeneity for FL over graph data, we firstly conduct preliminary studies on real-world data to investigate *whether different link-types should be treated differently in both global and federated training scenarios* (see subsection 3.3). Our preliminary results illustrate that (1) different link-types do preserve different levels of homophily (Table 2) and (2) the modeling of different link-types differently is beneficial in both global and federated learning scenarios (Table 3).

However, what are the necessary designs of an FL framework to properly differentiate and model the latent link-types across clients' local graphs? We decompose this into four sub-problems: P1: How to treat different link-types differently? We design a multi-channel GCN architecture (mGCN, see subsection 4.1) in which edges with different link-types will be modeled by different message passing channels; P2: How to train the mGCN model in the FL setting? On top of the FedAvg algorithm [21], we design a specialized aggregation algorithm for the split channels of mGCN that can aggregate model parameters of multiple link-type specific channels via a normalized averaging mechanism (Fed-mGCN, see subsection 4.2); P3: How to discover the link-types when they are latent? Motivated by the Expectation-Maximization (EM) algorithm [8] in k-means clustering [18], we design a clustered multi-channel GCN (cGCN, see subsection 4.3) that can automatically detect the link-types through the joint training of an edge clustering module and the message passing module; P4. How to train the cGCN model in the FL setting? To enable the link-type-oriented edge clustering through the collaboration of local clients, we design an FL algorithm in which not only the model parameters but also the cluster assignments are communicated and aggregated on the server. We name the proposed framework as dynamic latent **Li**nk-**T**ype aware clustered **Fed**erated graph learning, a.k.a. FedLit.

Since we are the first to study FL with graph data from the novel yet important perspective of latent link-type heterogeneity, we prepare our own datasets by introducing multiple link-types extracted from real-world data of academic networks and electronic health records (EHR), and then distribute the pre-processed datasets into varying numbers of clients via different ways of partitioning the link-types. We conducted extensive experiments on four datasets (DBLP-dm, PubMed-diabetes, NELL, MIMIC3) with three data partitions (distinct, dominant, balanced), in both centralized and FL settings. The results show that basic GCN and Fed-GCN cannot work well with latent link-type heterogeneity, while our framework is able to significantly improve both GCN and Fed-GCN. Furthermore, we conduct in-depth analysis from the perspectives of clustering effectiveness, pre-defined number of link-types, and clients' behaviors during FL.

Our work contributes in the innovation of both problem formulation and technique development:

- This is the first work to study the latent link-type heterogeneity problem in the setting of FL on graphs, including latent link-type heterogeneity across links, and across clients.
- We design a dynamic latent **Li**nk-**T**ype aware clustered **Fed**erated graph learning framework (FedLit) that can automatically detect the latent link-types underlying graphs and perform link-type-aware collaboration across clients.
- We build novel datasets with synthesized link-types from real-world data and conduct comprehensive experimental studies to demonstrate the superiority of our proposed framework and techniques.

## 2 RELATED WORKS

### 2.1 Federated Learning over Graph Data

The confluence of the rapid developing research on federated learning (FL) and emerging advanced graph neural networks (GNNs) promotes increased recent studies on FL over graph data. Prior works can be categorized into graph federated learning [2, 16, 23, 25] and federated graph learning [11, 17, 34, 35, 40]. Works in the former category models the FL architecture (the relations between servers and local devices, or among devices) by graphs and exploits graph models such as GNNs to facilitate FL, which has the focus on the FL setting. Closer to the latter category, we are motivated by the real FL scenarios with graph tasks as accessible applications, and focus on the unique challenges brought by graphs in the FL setting. Previous works of federated graph learning involve various topics, such as [22, 36, 41, 44] which focus on the privacy issue; [41, 44] which study the issue of isolated data island (cross-graph) and missing links; [4, 11, 34, 37, 42] which study the data heterogeneity (non-IID) problems w.r.t. graph structures and node attributes; and more application-oriented FL such as recommendation [36], knowledge graphs [6], and financial crimes detection [30].

Although some existing works have studied the data heterogeneity (non-IID) problems w.r.t. graphs in the FL setting, the graph specific non-IIDness across clients in FL is in fact rarely explored. [4] studies the heterogeneity and complementarity of graphs between clients for a global self-supervised FL framework, but it only considers the non-IIDness w.r.t. the numbers of nodes, edges, and labels, which are still the static sample-wise statistics of graphs and ignores the structural information (topology) of graphs. [37] studies the graph-level tasks across datasets and domains, in which the data heterogeneity w.r.t. graphs is at the graph-level and overlooks the

local neighborhood information. [11] proposed an FL benchmark with various GNN implementations, data partitions, and FL algorithms, while its non-IID data partitioning is still based on sample sizes (number of graphs). [34] leverages the model-agnostic meta-learning (MAML) method to learn a generative semi-supervised node classifier that can address the non-IID issue of graphs, but it does not focus on the graph heterogeneity regarding links.

## 2.2 Relation Learning without FL

As most GNNs have the assumption that similar nodes are more likely to be connected (homophily), they usually have difficulty in learning on heterophile graphs in which opposite nodes tend to connect. Although recently some works start to question whether the homophily is indeed necessary for GNNs [19, 20, 38], enormous research on heterophile graphs or graphs with varying heterophily keep emerging. Regarding the technical designs, these works can be categorized [43] into extension of neighborhoods [12, 13, 24] and dedicated designs of GNN architectures [5, 7, 45]. Beyond these specifically designed GNN models for heterophile graphs, [7] trains the GNN with a Generized PageRank algorithm regardless of the extent of homophily, and [12] claims that the real-world graphs are the composition of graphs with complete homophily/heterophily/randomness, and then chooses different hops of neighbors for different types of networks.

Apart from the homophily and heterophily of graphs, edges in real-world graphs can carry abundant semantic information, which can be regarded as relations. Specifically, relational learning can be applied to social networks for modeling the real-world social connections. For example, [39] learns the latent relations in social networks by leveraging the multi-modal semantic information, [31] incorporates relation learning to extract the latent social dimensions from the multi-dimensional connection in the social networks. [32] proposes a context-aware node embedding method to model the semantic relations between nodes. In our work, we include the semantic relations and the homophily/heterophily of graphs into a generalized concept named the latent link-types of graphs, and pioneer to study the unique challenges brought by the non-IID link-types across clients in the FL setting.

# 3 PROBLEM FORMULATION

## 3.1 Notations and Terms

We summarize all the notations that are used in this work in Table 1.

**Table 1: Notations and terms.**

| Notation | Representation |
|---|---|
| $G$ | A homogeneous graph. |
| $V; u, v$ | The set of nodes; $u$ and $v$ are nodes. |
| $E; e$ | The set of edges; $e$ is an edge. |
| $\mathbf{X}; \mathbf{y}$ | Node attributes; node labels. |
| $\mathbf{h}; \mathbf{z}$ | Node embedding; edge embedding. |
| $N; n$ | The number of clients; $n$ is the index. |
| $\pi$ | The number of oracle link-types. |
| $k$ | The number of pre-set number of clusters. |
| $\mathbb{C}; c$ | The set of link-types; $c$ is a link-type. |
| $\mathbb{C}^{\mathrm{k}}$ | The union set of link-types from all clients. |
| $\varphi^c$ | The centroid in the edge embedding space of link-type $c$. |
| $\tilde{\varphi}^c$ | The center of centroids $\{\varphi_n^c\}_{n\in N}$. |
| $\mathcal{D}_n(\psi; k)$ | The edge distribution of client $n$ w.r.t. $k$ link-types. |
| $\Theta; \theta^{\mathrm{s}}, \theta^{\boldsymbol{c}}, \theta^{\mathrm{t}}$ | The set of model parameters; parameters of different modules. |
| $r_{\mathrm{local}}; r$ | The local training epoch; the communication rounds. |

## 3.2 Preliminaries

*3.2.1 Graph convolutional network.* Graph convolutional network (GCN) [15] is a widely-used graph neural network that can be used to learn the representation of nodes by iteratively aggregating information propagated by their neighbors. Mathematically, given a graph $G = (V, E, \mathbf{X})$ with nodes $V$, edges $E$, and node attributes $\mathbf{X}$, a $l$-layer GCN can be formulated (in vector form) as

$$\mathbf{h}_u^{(l+1)} = \sigma\Big(\sum_{v\in\mathcal{N}_u} \frac{1}{\alpha_{uv}}\theta^{(l)}\mathbf{h}_v^{(l)}\Big), \tag{1}$$

where $\mathbf{h}_u^{(l+1)}$ is the representation of node $u$ at layer $l+1$, $v$ is a node from $u$'s neighbors $\mathcal{N}_u$, and $\theta$ is the learnable parameters of GCN. The normalization constant $\alpha_{uv}$ for an edge $e_{uv}$ is from the symmetrically normalized adjacency matrix $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, where $\tilde{A} = A + I$ is the summation of the adjacency matrix and the identity matrix, and $\tilde{D}$ is the diagonal node degree matrix of the matrix $\tilde{A}$.

*3.2.2 Federated averaging graph convolutional network (Fed-GCN).* FedAvg [21] is the most basic FL algorithm, which is based on stochastic gradient descent and aggregates the models uploaded by clients through normalized averaging. The vanilla FedAvg algorithm is applied to grid-structured data. Considering our setting of FL on graphs, we term the algorithm of training GCNs with FedAvg as Fed-GCN. Usually, FedAvg normalizes the aggregation by the sample sizes of clients. In Fed-GCN, when applying the node-level downstream task, e.g., node classification, we formulate the aggregation of Fed-GCN as

$$\theta^{(r+1)} = \sum_{n=1}^{|N|} \frac{|V_n|}{|V|}\theta_n^{(r)}, \tag{2}$$

where $\theta^{(r+1)}$ is the aggregated model at the $(r+1)^{\text{th}}$ communication round. $N$ represents the set of clients, $V_n$ is the node set of a graph on client $n$, and $\theta_n$ is the local uploaded model from client $n$. The aggregated model $\theta^{(r+1)}$ is then broadcast back to all clients.

## 3.3 Motivating Data Analysis

Given a graph $G = (V, E, \mathbf{X}, \mathbf{y})$ with $\pi$ known latent link-types, we would like to investigate whether it is necessary and beneficial to treat different link-types separately. We design metrics to quantitatively evaluate the latent link-types, and conduct preliminary experiments to compare the performance of node classification on datasets with different latent link-types, when using vanilla GCNs and multi-channel GCNs (see subsection 4.1) in both centralized and FL settings.

To have a straightforward comparison between link-types, we quantitatively evaluate them from two perspectives, Edge Homophily and Attribute Homophily. Edge Homophily (EH) [46] measures the edge-level homophily ratio w.r.t. node labels, which is the fraction of edges in the whole graph that connect nodes belonging to the same class, formulated as below

$$EH = \frac{|e_{uv} \in E \wedge y_u = y_v|}{|E|}. \tag{3}$$

Edges with various link-types could have a particular tendency of connecting nodes with the same labels or different labels as studied

in [12]. Thus, if EH diverges w.r.t. the generation rules underlying each link-type, it indicates that these links should be modeled separately. Beyond labels, we design another metric to evaluate the correlation between link-types and node attributes, termed as Attribute Homophily (AH). AS measures the mean homophily on attributes of all node pairs in a graph, formulated as

$$AH = \frac{\sum_{e_{uv} \in E} \mathrm{S}_{\cos}(\mathbf{x}_u, \mathbf{x}_v)}{|E|}, \tag{4}$$

where $\mathrm{S}_{\cos}$ is the cosine similarity. Edges with different link-types may particularly focus on certain attributes, and connect nodes with strong correlations between these attributes. Thus, if AH diverges w.r.t. the latent link-types, it also indicates the necessity to model the link-types separately.

To simulate the various latent link-types, we extract four potential relations as oracle link-types between papers (nodes) from a publication dataset DBLP-DM (see subsection 5.1), including: *reference*, *share authors*, *share keywords*, and *same year*. The *same year* link-type is generated on purpose for introducing some less relevant edges, which are common in real scenarios. By selecting the common nodes shared by all types of links, we construct four subgraphs, each of which has one distinct link-type, and one mixed subgraph that contains all edges of four link-types, all subgraphs with *research topics* as node labels. As displayed in Table 2, the values of AH and EH indicate that the characteristics of these link-types are indeed different. The *reference* link-type which should reflect the most direct relevance between papers shows the highest AH and EH, and the *same year* link-type which involves less relevant connections between papers shows the least AH and EH. The *share authors* and *share keywords* link-types are in-between. Furthermore, we would like to understand how such different characteristics of link-types as reflected by the AH and EH scores can influence the behaviors and utilities of graph learning models.

**Table 2: Edge Homophily and Attribute Homophily on subgraphs of DBLP-DM with distinct or mixed link-types.**

| Link-type | *reference* | *share authors* | *share keywords* | *same year* | mixed |
|---|---|---|---|---|---|
| EH | 0.2692 | 0.1847 | 0.2338 | 0.1150 | 0.1773 |
| AH | 0.2846 | 0.2394 | 0.2422 | 0.2044 | 0.2267 |

To experimentally valid the assumption that link-types can impact the performance of graph learning models, we apply the widely used GCN [15] and our intuitively designed multi-channel GCN and clustered multi-channel GCN (m-GCN and c-GCN, details in section 4) which can jointly model multiple latent link-types of the five subgraphs. From Table 3, the performance of GCN models varies on subgraphs with different link-types. When EH and AH decrease, the power of GCN models is impaired. Especially, for the subgraph with the least relevant edges that connect papers published in the same year, the performance of all models downgrades greatly. It is also noticed that the vanilla GCN cannot work well on the mixed graph with multiple link-types; however, mGCN can obviously improve from the vanilla GCN by separating the information propagation for different link-types. Additionally, cGCN is able to approach mGCN on the mixed graph without knowing the true latent link-types, and outperforms mGCN on graphs with single link-types, due to its ability to automatically uncover potential link-types in a data-driven fashion.

In the simplified illustrative simulation of the FL setting, each client owns a subgraph with a single link-type, and both the locally trained and collaboratively trained models are evaluated on a testing graph with mixed link-types. In the FL rows of Table 3, Fed-mGCN improves Fed-GCN by a large margin due to the separate handling of known different link-types. FedLit (Fed-cGCN), without knowing the ground-truth latent link-types, can still outperform Fed-GCN.

**Table 3: Centralized and FL performance (accuracy) w.r.t. subgraphs of DBLP-DM with distinct or mixed link-types.**

| Link-type | reference | share authors | share keywords | same year | mixed |
|---|---|---|---|---|---|
| GCN | 0.4066 | 0.3209 | 0.4159 | 0.1862 | 0.3403 |
| mGCN | 0.4025 | 0.3202 | 0.4155 | 0.1828 | **0.4017** |
| cGCN | 0.4177 | 0.3637 | 0.4315 | 0.2904 | 0.3797 |
| Fed-GCN | 0.3280 | | | | N/A |
| Fed-mGCN | **0.4125** | | | | N/A |
| FedLit | 0.3641 | | | | N/A |

## 3.4 Problem Setup

Given a server and $N$ local clients, each client holds a graph $G_n = (V_n, E_n, \mathbf{X}_n, \mathbf{y}_n)$. Suppose there are $\pi$ link-types underlying all clients' graphs $\{G_n\}_{n=1}^N$, then the graph on each client has the edge distribution w.r.t. $\pi$ latent link-types as

$$E_n \sim \mathcal{D}_n(\psi; \pi),$$

where $\psi$ is the parameters of data generating distribution for $\pi$ link-types. The set of link-types of $E_n$ on client $n$, denoted as $\mathbb{C}_n$, can have its size $|\mathbb{C}_n| \leq \pi$. An edge $e_{uv}^c \in E_n$ represents an edge of the graph $G_n$ on client $n$ that connects node $u$ and $v$ and has the link-type $c \in \mathbb{C}_n$.

On each graph $G_n$, the downstream task is node classification which predicts the label $y_u$ of node $u$ using the function $f(\Theta_n; G_n) : \mathbf{X}_n \rightarrow \mathbf{y}_n$, and $\Theta_n$ is the learnable parameters of function $f(\cdot)$. A client $n$ tries to find the optimized $\Theta_n^*$ by minimizing the local empirical risk

$$\mathcal{R}_n(f(\Theta_n; G_n)) \coloneqq \mathbb{E}[\ell(f(\Theta_n; G_n(u)), y_u)] \tag{5}$$

during its local training, where $\ell(\cdot)$ is the loss function such as cross entropy. The goal of the federated node classification framework over graphs is then to find $\Theta^*$ by

$$\Theta^* = \arg\min \mathcal{R}(f(\Theta; \{G_n\}_{n=1}^N)), \tag{6}$$

$$\mathcal{R}(f(\Theta; \{G_n\}_{n=1}^N)) \coloneqq \mathbb{E}_{n \in N}[\mathcal{R}_n(f(\Theta_n; G_n))]. \tag{7}$$

# 4 PROPOSED FRAMEWORK: FEDLIT

This work aims to resolve the latent link-type heterogeneity problems in FL on graphs, where clients can hold graphs with different latent link-types, or with the same latent link-types falling in different distributions. To approach it, we propose a dynamic latent **L**ink-**T**ype-aware clustered **Fed**erated graph learning framework (FedLit) that can automatically detect the link-types by dynamically clustering the edges, and perform local link-type-wise information propagation and global link-type-wise collaborative training.

Figure 2 shows an overview of our proposed framework FedLit. The framework consists of two main parts: I. the clustered multi-channel GCN (cGCN, subsection 4.3) which extends a multi-channel GCN (mGCN, subsection 4.1) with a co-trained clustering module;

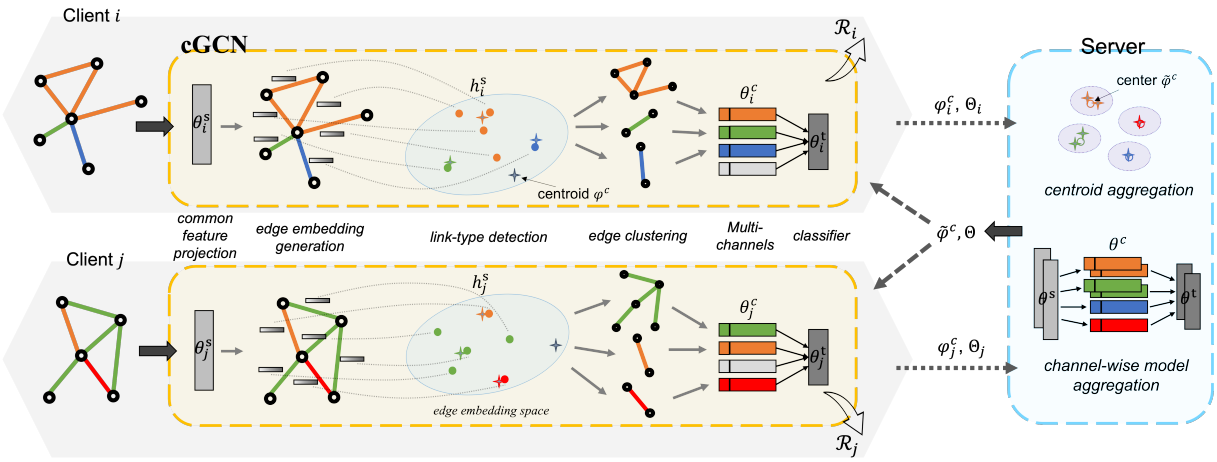



**Figure 2: Overview of FedLit: On each client, a local cGCN model with the edge embedding and clustering modules is iteratively trained based on its local graph and communicated across all clients through the server to achieve federated training.**

and II. FL of cGCN (subsection 4.4) which extends the static FL of mGCN (subsection 4.2).

## 4.1 The mGCN: multi-channel GCN

In order to distinguish the information propagation along different types of links, the graph model architecture at both local and global levels is designed as a multi-channel graph convolutional network (mGCN). The channels are implemented as an R-GCN [27] in this work, but other general multi-channel GCNs such as [33] can also be adapted. Specifically, it preserves a common feature projection layer shared by all link-types, multiple link-type-specific graph convolution channels, and a task-specific classifier.

The common feature projection layer with parameters $\theta^{\mathrm{s}} = [\vartheta_0^{\mathrm{s}}, \vartheta^{\mathrm{s}}]$ takes the original graph as an input, and outputs the node embedding by

$$\mathbf{h}_u^{\mathrm{s}} = \sigma(\vartheta_0^{\mathrm{s}} + \sum_{u \in V} \vartheta_u^{\mathrm{s}} \mathbf{x}_u). \tag{8}$$

On top of the common feature projection layer, the network is split into $k$ GCN channels and each is responsible for one link-type $c$, with parameters denoted as $\theta^c$. A channel for the link-type $c$ consists of a link-type-specific feature projection layer with parameters $\theta^{c_{\mathrm{s}}} = [\vartheta_0^{c_{\mathrm{s}}}, \vartheta_u^{c_{\mathrm{s}}}]$ and multiple graph convolution layers with parameters $\theta^{c_{\mathrm{c}},(l)}$ for layer $l$. The link-type-specific feature projection layer takes $\mathbf{h}_u^{\mathrm{s}}$ as the input for a node $u$ who has connected edges of link-type $c$, and outputs

$$\mathbf{h}_u^{c_{\mathrm{s}}} = \sigma(\vartheta_0^{c_{\mathrm{s}}} + \sum_{u \in V} \vartheta_u^{c_{\mathrm{s}}} \mathbf{h}_u^{\mathrm{s}}). \tag{9}$$

$\mathbf{h}_u^{c_{\mathrm{s}}}$ is then used for $l$ layers of graph convolution in a channel corresponding to the link-type $c$, and the output will be

$$\mathbf{h}_u^{c_{\mathrm{c}},(l+1)} = \sigma(\sum_{v \in \mathcal{N}_{u,e_{uv}=c}} \frac{1}{\alpha_{uv}} \theta^{c_{\mathrm{c}},(l)} \mathbf{h}_v^{c_{\mathrm{c}},(l)}), \tag{10}$$

where $\mathbf{h}_v^{c_{\mathrm{c}},(0)} = \mathbf{h}_v^{c_{\mathrm{s}}}$, and $v$ is a node from $u$'s neighbors $\mathcal{N}_{u,e_{uv}=c}$ connected by an edge of link-type $c$. $\alpha_{uv}$ is a normalization constant for the edge $e_{uv}$, which is computed from the symmetrically normalized adjacency matrix of graph $G$.

Since a node $u$ can be connected by multiple edges of different link-types, the embedding of $u$ is aggregated by its output embeddings generated from different channels, as long as there exists an edge of link-type $c$ connecting node $u$,

$$\tilde{\mathbf{h}}_u^c = \underset{c \in \mathbb{C}, \exists e_{u\cdot}=c}{\mathrm{agg}} (\mathbf{h}_u^{c_{\mathrm{c}},(l+1)}), \tag{11}$$

where $\mathrm{agg}(\cdot)$ can be either summation or averaging.

The last layer of a mGCN is a task-specific classifier that trains parameter $\theta^{\mathrm{t}} = [\vartheta_0^{\mathrm{t}}, \vartheta^{\mathrm{t}}]$ with a supervised downstream task such as node classification. The final prediction of node $u$ is calculated as

$$\mathbf{h}_u^{\mathrm{t}} = \sigma(\vartheta_0^{\mathrm{t}} + \sum_{u \in V} \vartheta_u^{\mathrm{t}} \tilde{\mathbf{h}}_u^c), \tag{12}$$

$$\hat{y}_u = \mathrm{softmax}(\mathbf{h}_u^{\mathrm{t}}). \tag{13}$$

We calculate the loss function using cross-entropy as

$$\ell(\Theta; G) = -\sum_{u \in V, label} y_u \log \hat{y}_u, \tag{14}$$

where $\Theta$ is the set of parameters $\theta^{\mathrm{s}}$, $\theta^c$, and $\theta^{\mathrm{t}}$ from modules of common feature projection s, multi-channel graph convolution $c$, and task-specific classifier t, respectively.

## 4.2 Fed-mGCN: federated learning of mGCNs

After the clients train mGCN for $r_{\mathrm{local}}$ epochs, they will upload their models $\Theta_n$ to the server for communication. Different from the FL of vanilla GCNs, the FL of mGCNs requires a dedicated design for model aggregation. First, for the common feature projection and the task-specific classifier, Fed-mGCN exploits the FedAvg algorithm [21] and normalized averaging for parameter aggregation $\{\theta_n^{\mathrm{s}}\}_{n \in N}$ and $\{\theta_n^{\mathrm{t}}\}_{n \in N}$ of all clients' models, respectively, by

$$\theta^{\mathrm{s},(r+1)} = \sum_{n=1}^{|N|} \frac{|V_n|}{|V|} \theta_n^{\mathrm{s},(r)}, \theta^{\mathrm{t},(r+1)} = \sum_{n=1}^{|N|} \frac{|V_n|}{|V|} \theta_n^{\mathrm{t},(r)}, \tag{15}$$

where $|V|$ is the total number of node samples of all clients, and $r$ is the communication round.

However, for split channels in mGCNs, the aggregation is in a manner that only channels learning information propagation

with the same link-type will be aggregated together. Here, the channels of mGCNs corresponding to link-type $c$ from all clients are normalized averaging aggregated by its own sample size, as

$$\theta^{c,(r+1)} = \sum_{n=1}^{|\{n\}_{\exists c \in \mathbb{C}_n}|} \frac{|V_n^c|}{|V^c|} \theta_n^{c,(r)}, \tag{16}$$

where $\{n\}_{\exists c \in \mathbb{C}_n}$ is the set of clients that have link-type $c$ in its graph $G_n$. That is, if $|\mathbb{C}_n| < k$, meaning that the number of link-types in $G_n$ is smaller than the total number of link-types $k$, only the channels responsible for link-type $c \in \mathbb{C}_n$ will contribute to the global model for collaboration. Besides, the aggregation is normalized by the sample size corresponding to link-type $c$ on client $n$ ($|V_n^c|$) over the total sample size $|V^c|$. Thus, clients with more edges of link-type $c$ can dominate the contribution of learning the optimal corresponding channels of a global mGCN.

### 4.3 The cGCN: clustered multi-channel GCN

With mGCN, we can realize the link-type-wise information propagation. However, regarding the fact that the link-types of graphs are unknown, how could we detect the link-types of edges for link-type-aware channel assignments? Conceptually, the detecting and clustering can happen either offline or online. The offline manner would finish the link-type detecting and edge clustering before transmitting the data into the GCN model for a downstream task. However, this way is practically infeasible since the link-types are implicit underlying graphs, and we do not know what attribute and structure information is essential for clustering edges that can benefit the performance. Hereby, we propose an online clustering method along with mGCN training, which can dynamically detect the latent link-types of edges and cluster edges.

Specifically, we represent edges by concatenating their two-end nodes' embeddings $\mathbf{h}^{\mathrm{s}}$ (Equation 8) from the common feature projection module of a mGCN, $\mathbf{z}_{uv} = \mathbf{h}_u^{\mathrm{s}} \oplus \mathbf{h}_v^{\mathrm{s}}$. The edge clustering is motivated by the k-means clustering [18], which exploits the expectation–maximization (EM) algorithm [8] and iteratively finds the centroids and minimizes the within-cluster distances. Given a predefined total number of link-types $k$, the initial $k$ centroids $\{\varphi^c\}_{c=1}^k$ are selected on the edge embedding space $\mathcal{Z}$ using k-means++ [1] for initialization. An edge $e_{uv}$ is then assigned to the cluster for link-type $c$ if it has the shortest distance from $\varphi^c$,

$$\Gamma(e_{uv}) \rightarrow c : \arg\min_c \mathrm{S}_{\cos}(\mathbf{z}_{uv}, \varphi^c), \tag{17}$$

where $\Gamma$ is the cluster assignment (M step). Within each cluster, the centroid $\varphi^c$ can be updated by an E step,

$$\varphi^c = \frac{1}{|\{e_{uv}\}_{e_{uv}=c}|} \sum_{e_{uv}=c} \mathbf{z}_{uv}. \tag{18}$$

Once a centroid $\varphi^c$ is initialized, it will fix its mapping to a channel in the mGCN architecture. After cGCN detects the link-types of edges and clusters edges accordingly, the subgraph with edges of link-type $c$ will be transmitted into the channel which is bound with the centroid $\varphi^c$. In each new training epoch, cGCN will re-run the E-M steps to detect edges' link-types and cluster edges, and then update the $k$ centroids. Herewith, the cluster assignment is co-trained with the model. In addition, since the centroids are selected in the edge embedding space $\mathcal{Z}$, and the edge embeddings $\mathbf{z}$ are generated from the model trained for downstream tasks, the link-type detection and edge clustering procedures are aware of the downstream task need.

### 4.4 FedLit: a dynamic latent link-type-aware clustered FL framework

In our proposed framework FedLit, each client $n$ holds a local model of cGCN, and preserves $k$ centroids $\{\varphi_n^c\}_{c \in \mathbb{C}^k}$. Because the edge cluster assignments are co-trained along the model training for downstream tasks, in the communication between client $n$ and the server, both the cGCN model parameters $\Theta_n$ of client $n$ and the set of effectually updated centroids $\{\varphi_n^c\}_{c \in \mathbb{C}_n}$ are transmitted for collaborative learning.

After model training for $r_{\text{local}}$ epochs, clients upload their models $\{\Theta_n\}_{n \in N}$ and centroids $\{\varphi_n\}_{n \in N}$ to the server. For these received $|N| * k$ centroids, the server will first separate them into $k$ groups $\{\phi_1, \phi_2, \ldots, \phi_k\}$, each with $|N|$ centroids from distinct clients, $\phi_c = \{\varphi\}_n$, so that a group $\phi_c$ is corresponding to a link-type $c$. The objective is to find a grouping that minimizes the within-group summation of distances,

$$\arg\min_{\phi} = \sum_{i=1}^{k} \sum_{\varphi \in \phi_c} ||\varphi - \tilde{\varphi}_c||^2, \tag{19}$$

with the constraint that all $\varphi$ in $\phi_i$ should be from different clients. And $\tilde{\varphi}_c$ is the center of group $\phi_c$,

$$\tilde{\varphi}_c = \frac{1}{|\phi_c|} \sum_{\varphi \in \phi_c} \varphi. \tag{20}$$

With $\{\phi_c\}_{c=1}^k$, the server knows which channels of the cGCNs in clients are responsible for which link-types. Then the local centroids of clients $\varphi_n^c$ belonging to the group $\phi_c$ can be updated by $\tilde{\varphi}^c$.

Meanwhile, as the grouping maintains the alignments between local centroids and global centers, the correspondence between channels of local cGCNs among clients and channels of global cGCN in the server is also known. Therefore, the channel aggregation of cGCNs in FedLit follows the grouping $\{\phi_c\}_{c=1}^k$ where channels corresponding to $\varphi \in \phi_c$ will be aggregated and updated using Equation 16. The aggregation of common feature projection and task-specific classifier among clients are same as Equation 15.

After aggregation, the server broadcasts the updated model and centers to clients following the grouping. Herewith, the local cluster assignments are not only co-trained with its local model for downstream tasks, but also involved in the communication of FL for collaborating with other clients to discover consistent edge clusters and latent link-types globally.

## 5 EXPERIMENTS

### 5.1 Experimental Settings

*5.1.1 Data.* Since this is the first work to study latent link-types of graphs in the FL setting, we pre-process four public raw datasets and constructed four graphs with various oracle link-types. We include two publication datasets, DBLP-DM[1] and PubMed-diabetes[2], and

[1] DBLP-DM: https://dblp.uni-trier.de/

[2] PubMed-diabetes: https://linqs.org/datasets/#pubmed-diabetes

**Table 4: Accuracy on publication and medical datasets with three data partitions. Bold represents the best results in centralized and FL settings, respectively. Grey background represents the best results among all methods excluding mGCN and Fed-mGCN.**

| Dataset | DBLP-DM | | | PUBMED-DIABETES | | | NELL | | | MIMIC3 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GCN | 0.3378 (±0.0022) | | | 0.8468 (±0.0024) | | | 0.3758 (±0.0210) | | | 0.3603 (±0.0021) | | |
| cGCN [$k = \pi$] | 0.3712 (±0.0042) | | | 0.8781 (±0.0055) | | | 0.4983 (±0.0198) | | | 0.3655 (±0.0039) | | |
| cGCN [$k > \pi$] | 0.3884 (±0.0048) | | | **0.8795** (±0.0028) | | | 0.5024 (±0.0069) | | | 0.3715 (±0.0030) | | |
| mGCN | **0.4148** (±0.0026) | | | 0.8778 (±0.0023) | | | **0.5162** (±0.0129) | | | **0.3973** (±0.0028) | | |
| | distinct | dominant | balanced | distinct | dominant | balanced | distinct | dominant | balanced | distinct | dominant | balanced |
| Fed-GCN | 0.3033 (±0.0028) | 0.3088 (±0.0025) | 0.3226 (±0.0036) | 0.7776 (±0.0046) | 0.8008 (±0.0043) | 0.8284 (±0.0029) | 0.2345 (±0.0184) | 0.2260 (±0.0016) | 0.3491 (±0.0107) | 0.3399 (±0.0055) | 0.3388 (±0.0026) | 0.3584 (±0.0032) |
| Fed-mGCN | **0.3930** (±0.0016) | **0.3779** (±0.0050) | **0.4046** (±0.0015) | 0.8758 (±0.0025) | 0.8657 (±0.0058) | 0.8745 (±0.0022) | 0.4447 (±0.0178) | 0.4050 (±0.0055) | **0.5178** (±0.0065) | **0.3830** (±0.0015) | 0.3474 (±0.0020) | 0.3762 (±0.0011) |
| FEDLIT [$k = \pi$] | 0.3238 (±0.0049) | 0.3371 (±0.0066) | 0.3400 (±0.0083) | **0.8776** (±0.0044) | 0.8779 (±0.0028) | 0.8771 (±0.0070) | 0.4158 (±0.0071) | 0.3970 (±0.0067) | 0.4750 (±0.0152) | 0.3552 (±0.0017) | 0.3541 (±0.0037) | 0.3750 (±0.0048) |
| FEDLIT [$k > \pi$] | 0.3533 (±0.0040) | 0.3701 (±0.0080) | 0.3669 (±0.0062) | 0.8759 (±0.0092) | **0.8820** (±0.0047) | **0.8811** (±0.0023) | **0.4715** (±0.0111) | **0.4243** (±0.0253) | 0.5091 (±0.0060) | 0.3685 (±0.0018) | **0.3593** (±0.0038) | **0.3765** (±0.0046) |

two electronic health record (EHR) datasets, NELL[3] and MIMIC3[4]. In general, for each raw dataset, we sample data, generate node features, create node labels, and construct links using different link generation rules. For DBLP-DM, the oracle link-types are defined as: i) references between papers, ii) papers sharing author(s), iii) papers sharing more than one keyword, and iv) papers published in the same year. PUBMED-DIABETES has the same oracle link-types i), ii), and iv) as DBLP-DM, and the link-type iii) that papers appear among the top-k similar papers of others. For the EHR datasets, their oracle link-types include: i) admissions of the same patients, ii) the overlap ratios of procedures between two admissions above a certain threshold, iii) the overlap ratios of prescription between two admissions above a certain threshold, and iv) the time frames of stay between two admissions are overlapped. The statistics of constructed graphs with link-types are shown in Table 5.

**Table 5: Statistics of processed publication and EHR datasets.**

| Dataset | #Node | #Feature | #Class | #Total Edge | #Oracle Edge i) | ii) | iii) | iv) |
|---|---|---|---|---|---|---|---|---|
| DBLP-DM | 46,582 | 200 | 12 | 7,097,924 | 206,219 | 1,285,315 | 2,818,120 | 2,788,270 |
| PUBMED-DIABETES | 13,778 | 200 | 3 | 588,529 | 20,035 | 118,441 | 74,971 | 375,082 |
| NELL | 41,671 | 2,792 | 5 | 39,250,315 | 91,229 | 6,499,135 | 24,529,421 | 8,130,530 |
| MIMIC3 | 58,495 | 6,671 | 6 | 30,603,469 | 23,068 | 27,244,566 | 2,413,231 | 922,604 |

#### 5.1.2 *Settings.*

*Partitioning.* We consider three ways of data partitioning, in which each simulates a different extent of link-type heterogeneity across clients. a) Distinct. It simulates the situation in FL where clients preserve graphs with different link-types, in which there exists link-type heterogeneity w.r.t. the set of link-types across clients. We extremize the situation by making a client hold only one link-type. Each client randomly chooses a link-type and receives a subgraph with the link-type sampled from the full graph. b) Dominant. It simulates a more natural situation in FL where clients preserve graphs with all link-types but have a dominant link-type of their interests. In this situation, there exists the link-type heterogeneity w.r.t. link-type distribution across clients. Each client randomly chooses a link-type of its interest, and receives a subgraph sampled from the full graph which contains most edges with the focused (dominant) link-type and smaller portions of edges with other link-types. c) Balanced. It simulates an unnatural situation in FL where clients preserve graphs with minimum link-type heterogeneity. In this situation, the sampled graphs on clients have the exactly same distribution of edges with different link-types.

*Baselines.* We design baselines in the centralized learning setting, including a) GCN which trains a vanilla GCN [15] with feature projection layers and a classifier, b) mGCN which trains a multi-channel GCN on a graph with oracle link-types, and c) cGCN which trains an mGCN on a graph without knowing its oracle link-types, and in the FL setting, including d) Fed-GCN which aggregates the vanilla GCN models on local clients by the FedAvg algorithm, and e) Fed-mGCN which aggregates the mGCN models on local clients in a channel-wise manner.

*Hyper-parameters.* The architecture of all GNN models includes two feature projection layers, two graph convolutional layers, and a classifier layer. We use Adam [14] optimizer with a learning rate of 0.01 for the publication datasets, and with a learning rate of 0.001 for the EHR datasets. The numbers of training epochs and communication rounds are 100. In FL settings, the number of clients is 10, and the local training epoch $r_{\text{local}}$ is set as 1. We run 10-fold cross-validation for each experiment. All experiments are run on a server with eight 48GB NVIDIA Quadro RTX 8000 GPUs. All code and data are provided in this GitHub repository[5].

## 5.2 Experimental Results at Global Level

Table 4 displays the comprehensive results of evaluating all methods on a global graph in the centralized and FL settings. In general, FEDLIT can outperform all FL baselines in all cases except for Fed-mGCN, due to its access to oracle link-types. In the centralized learning setting, the cGCN model leveraging our designed clustering module can get very close to the mGCN model with access to oracle link-types. In the FL setting, our framework FEDLIT achieves comparable results to the corresponding centralized cGCN model, and consistently beats the Fed-GCN model. Conceptually, the baseline Fed-mGCN should perform best in the FL setting. However, the oracle link-types are not always the optimal ground truth (the links can still be heterogeneous even with the same oracle types), and the intentionally introduced noisy edges with the link-type that is less relevant to the tasks may deteriorate the rigid framework of mGCN. Thus, it can be observed from the table that half of the results of FEDLIT actually surpass the results of Fed-mGCN.

Although the "oracles" of link-types may not be ideal, we still use them as the ground-truth link-types and experiment our designed clustering module with the number of channels equal to ($k = \pi$) or greater than ($k > \pi$) the number of oracle link-types. It is observed that the results of cGCN and FEDLIT with [$k > \pi$] are always better

[3]NELL: https://www.nursing.emory.edu/pages/project-nell
[4]MIMIC3: https://physionet.org/content/mimiciii-demo/1.4/
[5]https://github.com/Oxfordblue7/FedLIT

than the results with $[k = \pi]$, except for the distinct setting on PubMed-diabetes where FedLit with $[k = \pi]$ slightly outperforms that with $[k > \pi]$. This observation suggests that simply setting the number of clusters $k$ to be larger than the real number of link-types can often guarantee satisfactory performance.

Regarding the different data partitioning in the FL setting, Fed-GCN often performs better in the balanced partitioning setting than in the distinct and dominant partitioning settings, which is natural because the balanced partitioning setting is closest to the IID case which is easiest for the vanilla FedAvg algorithm in Fed-GCN. Meanwhile, it further proves that FL with vanilla GCNs would fail when facing significant link-type heterogeneity. However, with multiple-channel GCNs, i.e., Fed-mGCN and FedLit, the distinct and dominant partitioning settings have the chance to surpass the balanced partitioning setting, which indicates that our proposed frameworks are able to address the link-type heterogeneity problem effectively. Additionally, the larger the extent of link-type heterogeneity is (i.e., distinct > dominant > balanced), the effects of our proposed framework FedLit are more compelling, as demonstrated by the larger gains over Fed-GCN.

## 5.3 Experimental Results at Local Level

**Table 6: Accuracy on local clients with subgraphs of DBLP-dm from two data partitions. Bold represents the best results and grey background indicates results from our framework.**

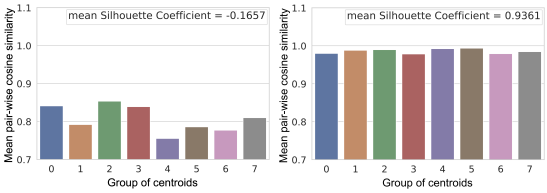
| Partition | distinct | | | dominant | | |
|---|---|---|---|---|---|---|
| Client | n7 | n8 | n9 | n7 | n8 | n9 |
| GCN | 0.1828 (±0.0023) | 0.1856 (±0.0050) | 0.1862 (±0.0023) | 0.2745 (±0.0029) | 0.2708 (±0.0027) | 0.2759 (±0.0022) |
| Fed-GCN | 0.1771 (±0.0013) | 0.1770 (±0.0003) | 0.1801 (±0.0011) | 0.2721 (±0.0014) | 0.2746 (±0.0015) | 0.2784 (±0.0016) |
| Fed-mGCN | 0.1839 (±0.0029) | 0.1842 (±0.0042) | 0.1850 (±0.0035) | 0.3462 (±0.0046) | 0.3392 (±0.0031) | 0.3408 (±0.0018) |
| FedLit | **0.3126** (±0.0072) | **0.3115** (±0.0057) | **0.3170** (±0.0054) | **0.3579** (±0.0041) | **0.3590** (±0.0051) | **0.3565** (±0.0031) |

As noisy edges with link-type less relevant to the downstream tasks are introduced into the graphs, it is motivating to investigate whether the clients with most noisy edges would benefit from FL using our framework FedLit. Table 6 demonstrates local results of three clients (i.e., n7, n8, n9) with the same link-type (i.e., link-type iv)). Overall, FedLit can significantly improve the performance of local clients compared to training GCNs on the clients themselves, by a large margin of 13% in distinct partitioning and 8% in dominant partitioning settings. However, Fed-GCN has no effect on helping the lagging clients. Fed-mGCN can improve the local clients in the dominant partitioning setting because of the channel-wise collaboration. While in the distinct partitioning setting, the performance of Fed-mGCN is similar to the local training because channels without input edges will not participate in the collaboration.

# 6 IN-DEPTH ANALYSES

## 6.1 Clustering Analysis

We delve into the edge clustering w.r.t link-types of FedLit to investigate whether the communication among local centroids is effective. To address the question, we focus on the groups of centroids corresponding to different link-types on the server and evaluate the within-group similarity of the centroids. If the edge clustering module w.r.t. link-types effectively communicates, the within-group similarity of centroids should increase during FL. The grouped bar plot in Figure 3 displays the mean pair-wise cosine similarity between centroids within groups, in which each bar represents a group corresponding to an edge cluster assignment. According to the bar plots at the first and last communication rounds, it is obvious that the mean pair-wise cosine similarity of each group increases after training of FedLit. Especially, the mean pair-wise cosine similarity of all groups approaches 1.0 at round 100. In addition, we also measure the Silhouette coefficients [26] for each group, evaluating the goodness of a cluster assignment and averaging them over all groups to get the mean Silhouette coefficient. We observe a high mean Silhouette coefficient over groups which is very close to 1 at round 100. These results indicate that the edge clustering w.r.t. link-types on clients in our proposed framework FedLit are effectively communicated and trained across clients.

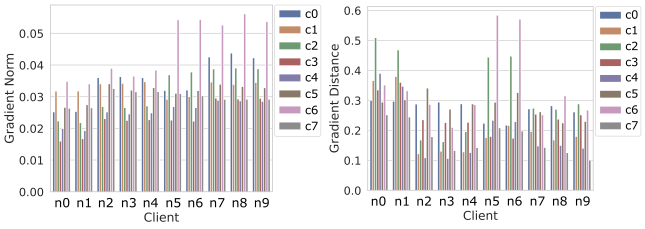



**(a) round 1** **(b) round 100**

**Figure 3: Clustering analysis results on DBLP-dm.**

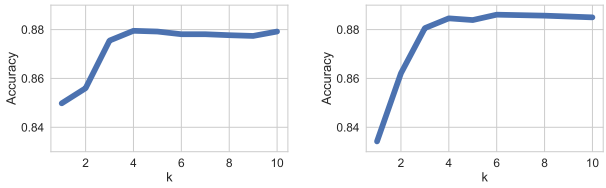
## 6.2 Client Behavior Analysis

We delve into the behaviors of multiple channels of cGCN on local clients during FL with FedLit, to investigate whether the channels can be effectively learned for different link-types. We consider two points: whether the behaviors of multiple channels are different, and whether the clients with similar link-type distribution show similar behaviors w.r.t. the channels. To measure these behaviors, we compute two metrics, (a) gradient norm, and (b) gradient distance. The gradient norm of a channel is calculated by averaging its norms of gradients over the communication rounds. The gradient distance of a channel is calculated by averaging the cosine distances between its gradients and the server aggregated gradients over the communication rounds. Figure 4 visualizes the gradient norm and gradient distance of each channel (c*) in local models on all clients (n*). The grouped bars for each client can be regarded as a distribution of the channel's behaviors during FL. Thus, we can infer which clients preserve similar link-type distributions. According to the data partitioning, clients n0 and n1 have the dominant oracle link-type i); client n2, n3, and n4 have the dominant oracle link-type ii); client n5 and n6 have the dominant oracle link-type iii); and client n7, n8, and n9 have the dominant oracle link-type iv). In both Figure 4a and 4b, it can be observed that clients with the same dominant link-type show similar distribution of gradient norm and gradient distance. In addition, the gradient distance measures the channel-wise contribution of clients to the global model. From Figure 4b, we can conclude that the c6 channel of models on clients n5 and n6, and the c2 channel of models on clients n0, n1, n5, and n6, contribute the most to the global model.




(a) Gradient Norm (b) Gradient Distance

**Figure 4: FL analysis results on DBLP-DM**




(a) cGCN (b) FEDLIT

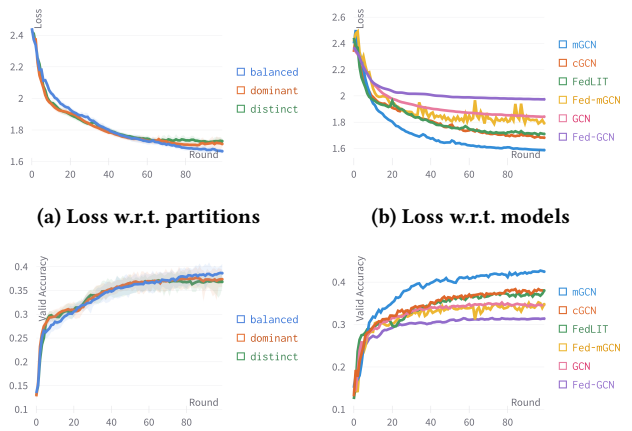
**Figure 5: Hyper-parameter analysis results.**

## 6.3 Hyper-parameter Analysis

The number of link-type $k$ should be pre-defined as a hyper-parameter and can be tuned to achieve better performance. To study the effect of the hyper-parameter $k$ and find the rule-of-thumb for choosing a reasonable $k$, we analyze the performance of cGCN and FEDLIT with four oracle link-types when varying $k$. As shown in Figure 5, increasing $k$ till a certain value will benefit the performance of cGCN and FEDLIT, but continuing increasing $k$ will not further improve the performance as the performance will converge. The reason can be that when $k$ is larger than the actual need of number of channels, multiple channels would be used to model one link-type, which should neither facilitate nor deteriorate the performance, but will cost more computational resources and runtime. With the observation, we can conclude the rules for choosing a reasonable $k$ as: 1) choosing the exact best $k$ is not necessary since the performance will converge as $k$ grows large; 2) one can choose a relatively larger $k$ that is affordable under the considerations of computational resources and runtime budgets.

## 6.4 Convergence Analysis

To analyze the convergence of FEDLIT, we visualize the training curves and validation accuracy w.r.t. three ways of data partitioning and w.r.t. all models in Figure 6. It can be seen that the validation accuracy has the correct correspondence to the training curves among all models, where the order from large loss to small loss is aligned with the order from low performance to high performance. From the training curves in Figure 6b, we conclude that our proposed framework can converge with similar speeds as all other baselines. By observing the validation accuracy in Figure 6d, our proposed framework FEDLIT can achieve comparable performance as cGCN in the centralized setting, and always outperform the baselines of GCN and Fed-GCN. Assuming the condition that the oracle link-types are accurate, our framework FEDLIT would naturally have a performance lower than mGCN and Fed-mGCN.




(a) Loss w.r.t. partitions (b) Loss w.r.t. models

(c) Accuracy w.r.t. partitions (d) Accuracy w.r.t. models

**Figure 6: Convergence analysis results on DBLP-DM**

## 7 CONCLUSION

This work focuses on FL on graphs with latent link-type heterogeneity. Specifically, it studies the problems that the latent link-types including the extent of homophile and semantic relations vary among local clients, and the distributions of latent link-types are non-independent and identical. To resolve the problems, we propose a dynamic latent link-type-aware clustered federated graph learning framework (FEDLIT) that automatically detects the latent link-types in graphs and performs link-type-aware FL. We synthesize multiple datasets from benchmark data sources to simulate the latent link-type heterogeneity w.r.t. link-types, and design different data partitioning to introduce varying levels of link-type heterogeneity among clients. The comprehensive experimental results on these datasets demonstrate the effectiveness of FEDLIT. Additionally, by investigating the performance of the federated learned global model on local clients, we found that our framework FEDLIT can greatly improve the lagging clients with most noisy edges by large margins. We further analyze the clustering of FEDLIT during communication, and found that the framework is able to collaboratively learn the cluster assignments effectively. To better understand the learning behavior of channels in FEDLIT, we investigate the channel-wise contribution of a client model to the aggregated model, and the analysis demonstrates that the channels are different and we can infer the groups of clients with similar link-type distributions. We also analyze the effects of the hyper-parameter $k$ and conclude the rule-of-thumb for choosing $k$, and finally observe the training curves and validation accuracy to investigate the convergence of FEDLIT.

One limitation of our framework is that the edge clustering can hardly generate empty clusters when necessary, because of the large edge embedding space. Although in most clustering situations one wants to avoid empty clusters, the empty clusters can be useful in our scenarios if there are indeed some link-types missing on a client. Thus, our framework may cluster some edges wrongly in such scenarios. The good side of this problem is that the wrongly clustered edges usually only occupy a small portion of the full set of edges and therefore the overall performance will not be strongly impeded. Another limitation of FEDLIT is that we focus less on

the efficiency consideration in this work, and the current training pipeline for FedLit takes observably longer time than Fed-GCN, although the good side is such longer training time is still linear regarding the size of graphs and training epochs, and FedLit usually does not need significantly more epochs to converge. In the future, we could further improve FedLit in the consideration of efficiency by involving some edge sampling techniques to approximate the edge clusters, and some model compression or model distillation techniques regarding the local model aggregations. We can further study the privacy and fairness perspectives of FedLit in the contexts of link heterogeneity, and the generalization of the spiritual framework of FedLit onto other types of data beyond graphs.

## REFERENCES


[1] David Arthur and Sergei Vassilvitskii. 2006. *k-means++: The advantages of careful seeding*. Technical Report.

[2] Debora Caldarola, Massimiliano Mancini, Fabio Galasso, Marco Ciccone, Emanuele Rodolà, and Barbara Caputo. 2021. Cluster-driven Graph Federated Learning over Multiple Domains. In *CVPRW*.

[3] Chen Chen, Jingrui He, Nadya Bliss, and Hanghang Tong. 2017. Towards optimal connectivity on multi-layered networks. *IEEE transactions on knowledge and data engineering* (2017).

[4] 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).

[5] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *ICML*.

[6] Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. 2021. FedE: Embedding Knowledge Graphs in Federated Setting. In *IJCKG*.

[7] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2020. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*.

[8] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)* (1977).

[9] Xu Han, Haoran Yu, and Haisong Gu. 2019. Visual inspection with federated learning. In *ICIAR*.

[10] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[11] Chaoyang He, Keshav Balasubramanian, Emir Ceyani, Carl Yang, Han Xie, Lichao Sun, Lifang He, Liangwei Yang, Philip S Yu, Yu Rong, et al. 2021. Fedgraphnn: A federated learning system and benchmark for graph neural networks. *arXiv preprint arXiv:2104.07145* (2021).

[12] Di Jin, Zhizhi Yu, Cuiying Huo, Rui Wang, Xiao Wang, Dongxiao He, and Jiawei Han. 2021. Universal Graph Convolutional Networks. In *NeurIPS*.

[13] Wei Jin, Tyler Derr, Yiqi Wang, Yao Ma, Zitao Liu, and Jiliang Tang. 2021. Node Similarity Preserving Graph Convolutional Networks. In *WSDM*.

[14] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. In *ICLR*.

[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[16] Anusha Lalitha, Osman Cihan Kilinc, Tara Javidi, and Farinaz Koushanfar. 2019. Peer-to-peer federated learning on graphs. *arXiv preprint arXiv:1901.11173* (2019).

[17] Rui Liu and Han Yu. 2022. Federated Graph Neural Networks: Overview, Techniques and Challenges. *arXiv preprint arXiv:2202.07256* (2022).

[18] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* (1982).

[19] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2021. Is Heterophily A Real Nightmare For Graph Neural Networks To Do Node Classification? *CoRR* (2021).

[20] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. 2022. Is Homophily a Necessity for Graph Neural Networks?. In *ICLR*.

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

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

[23] Chuizheng Meng, Sirisha Rambhatla, and Yan Liu. 2021. Cross-Node Federated Graph Neural Network for Spatio-Temporal Data Modeling. In *KDD*.

[24] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287* (2020).

[25] Elsa Rizk and Ali H. Sayed. 2021. A Graph Federated Architecture with Privacy Preserving Learning. In *SPAWC*.

[26] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* (1987).

[27] 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 *European semantic web conference*.

[28] Santiago Silva, Boris A Gutman, Eduardo Romero, Paul M Thompson, Andre Altmann, and Marco Lorenzi. 2019. Federated learning in distributed medical databases: Meta-analysis of large-scale subcortical brain data. In *ISBI*.

[29] Yizhou Sun and Jiawei Han. 2013. Mining heterogeneous information networks: a structural analysis approach. *Acm Sigkdd Explorations Newsletter* (2013).

[30] 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. *arXiv preprint arXiv:1909.12946* (2019).

[31] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *KDD*.

[32] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. CANE: Context-Aware Network Embedding for Relation Modeling. In *ACL*.

[33] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *ICLR*.

[34] 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).

[35] Chunnan Wang, Bozhou Chen, Geng Li, and Hongzhi Wang. 2021. FL-AGCNS: federated learning framework for automatic graph convolutional network search. *arXiv preprint arXiv:2104.04141* (2021).

[36] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv preprint arXiv:2102.04925* (2021).

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

[38] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. 2021. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. *arXiv preprint arXiv:2102.06462* (2021).

[39] Carl Yang, Jieyu Zhang, Haonan Wang, Sha Li, Myungwan Kim, Matt Walker, Yiou Xiao, and Jiawei Han. 2020. Relation Learning on Social Networks with Multi-Modal Graph Edge Variational Autoencoders.

[40] 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).

[41] Ke ZHANG, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. 2021. Subgraph Federated Learning with Missing Neighbor Generation. In *NeurIPS*.

[42] 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).

[43] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. 2022. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082* (2022).

[44] Jun Zhou, Chaochao Chen, Longfei Zheng, Huiwen Wu, Jia Wu, Xiaolin Zheng, Bingzhe Wu, Ziqi Liu, and Li Wang. 2020. Vertically federated graph neural network for privacy-preserving node classification. *arXiv preprint arXiv:2005.11903* (2020).

[45] Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. 2021. Graph neural networks with heterophily. In *AAAI*.

[46] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *NeurIPS*.