Federated Node Classification over Distributed Ego-Networks with Secure Contrastive Embedding Sharing

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

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

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

Abstract

Federated learning on graphs (a.k.a., federated graph learning– FGL) has recently received increasing attention due to its capacity to enable collaborative learning over distributed graph datasets without compromising local clients' data privacy. In previous works, clients of FGL typically represent institutes or organizations that possess sets of entire graphs (e.g., molecule graphs in biochemical research) or parts of a larger graph (e.g., sub-user networks of e-commerce platforms). However, another natural paradigm exists where clients act as remote devices retaining the graph structures of local neighborhoods centered around the device owners (i.e., ego-networks), which can be modeled for specific graph applications such as user profiling on social ego-networks and infection prediction on contact ego-networks. FGL in such novel yet realistic ego-network settings faces the unique challenge of incomplete neighborhood information for non-ego local nodes since they likely appear and have different sets of neighbors in multiple ego-networks. To address this challenge, we propose an FGL method for distributed ego-networks in which clients obtain complete neighborhood information of local nodes through sharing node embeddings with other clients. A contrastive learning mechanism is proposed to bridge the gap between local and global node embeddings and stabilize the local training of graph neural network models, while a secure embedding sharing protocol is employed to protect individual node identity and embedding privacy against the server and other clients. Comprehensive experiments on various distributed ego-network datasets successfully demonstrate the effectiveness of our proposed embedding sharing method on top of different federated model sharing frameworks, and we also provide discussions on the potential efficiency and privacy drawbacks of the method as well as their future mitigation.

CCS Concepts

• **Information systems** → *Data mining*; • **Computing methodologies** → **Distributed artificial intelligence**; *Neural networks*; Machine learning; • **Security and privacy** → **Privacy-preserving protocols**; • **Theory of computation** → Social networks.

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679834

Keywords

Federated Learning, Graph Neural Networks, Ego-Networks, Efficiency, Privacy, Contrastive Learning, Secure Sharing

ACM Reference Format:
Han Xie, Li Xiong, and Carl Yang. 2024. Federated Node Classification over Distributed Ego-Networks with Secure Contrastive Embedding Sharing. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3627673.3679834

1 Introduction

Federated learning (FL) is a distributed learning paradigm designed to facilitate collaborative model training without sharing local data, in the considerations of data privacy, difficulty in centralizing data, and lack of high-quality data to train promising local models (e.g., data scarcity, imbalance, and bias) [20, 31, 31, 57, 64]. Recently, FL with graph data, a.k.a., federated graph learning (FGL), has attracted a lot of attention due to the emergence of vast amounts of real-world graphs and their properties distinguished from general Euclidean data [9, 16, 27]. In FGL, graphs can be distributed in different ways according to real scenarios. For example, for molecule studies in the biochemical domain, local clients such as biomedical institutes possess sets of entire molecular graphs and aim for graph-level downstream applications like predicting the property of molecules [17, 28, 29, 66]. For healthcare, finance, or businesses, local clients such as hospitals, banks, or e-commerce platforms, usually possess only parts of a global patient or customer network and desire to collectively train models with distributed subgraphs for downstream graph tasks like node classification and link prediction [26, 41, 43, 51]. FGL over distributed subgraphs presents more challenges due to the information missing across clients and entity alignments among clients. Recent works of FGL have been devoted to resolving these issues [1, 13, 24, 58, 61, 63].

Apart from FGL across distributed sets of entire graphs and distributed subgraphs, there exists another essential real-world scenario where each client represents an individual or remote device and possesses an ego-network that only includes the neighborhood information centered on the ego itself, and many such clients want to collaboratively train better local models. This setting, i.e., *FGL over distributed ego-networks*, is realistic and crucial as it aims to protect users' personal information while enhancing the utilities of downstream applications. For example, in a heterogeneous e-commerce network of customers and sellers, user profiling is a critical application that can identify various user attributes such as habits and interests of customers towards better product recommendations. However, customers may not want their personal data such as browsing behaviors and purchasing histories to be

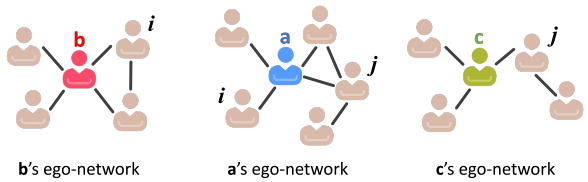


Figure 1: Toy examples of distributed ego-networks.

freely shared among different sellers for privacy concerns; herewith, the sellers (as clients) can aggregate secure versions of the missing data from other clients' ego-networks to achieve better local utilities for recommendation. A toy example can be seen from Figure 1– if the information of customer i is incomplete in seller a's ego-network while customer i also interacts with seller b, then seller a can aggregate information about customer i from seller b to improve his/her local modeling of customer i. Another real scenario is infection prediction in local contact networks. During pandemics, it is crucial that individuals can assess the infection risks in their contact networks to avoid certain contacts that may further spread the diseases. As also shown in Figure 1, individual a wants to assess the infection risk of individual j, but j also contacts with other individuals in individual c's ego-network. The assessment of a can thus benefit from information about j obtained from c.

Besides the generic challenges of FL such as local data scarcity, imbalance, and bias (which encourages the federated sharing of models across clients), FGL over distributed ego-networks faces an additional unique challenge of incomplete information of local nodes (which encourages the federated sharing of embeddings across clients). In the context of graph learning, the information of each local node can be incomplete regarding both their own node features and the features and link structures of their neighboring nodes. Thus, our key insight is, sharing models such as the graph neural networks (GNNs) cannot effectively bridge this gap of incomplete local neighborhood information. In Table 1, we empirically illustrate the impact of such incomplete information in distributed local ego-networks. We sample 100 nodes as local clients with two hops of ego-networks from the commonly used network datasets of Cora[30] and LastFM-Asia[35], and evaluate standard GCN models [23] trained with different mechanisms towards local node classification tasks. It is obvious that locally trained models (i.e., Local) perform worse than the globally trained model (i.e., Global), possibly due to insufficient local data samples or incomplete local neighborhood information. However, FL with model-sharing mechanisms (i.e., FedAvg [31] and FedProx [25]) has no effect in improving over the locally trained models, showcasing that simply increasing local data samples (training with more local data samples by engaging more clients in FL collaboration) is not beneficial and model-sharing cannot mitigate incomplete local neighborhood information.

Contributions. In this work, we propose Secure Contrastive Embedding Sharing (a.k.a., `FedSCem`) for federated node classification over distributed ego-networks. To bridge the gap of incomplete local neighborhood information in FL with distributed ego-networks, we propose to share node embeddings across clients. Analogous to model sharing based on FedAvg [31], this process involves clients uploading locally computed node embeddings to the server, and the server computing the globally aggregated node embeddings and sending them back to clients. Furthermore, we design a contrastive learning mechanism to bridge the gap between locally computed and globally aggregated embeddings during each FL iteration to stablize the local training of GNN models. To address the privacy issues of embedding sharing, we employ a secure sharing protocol and proves its capabilities of protecting important private information regarding individual node identities and embeddings in each local client against the server and other clients. Finally, we also provide comprehensive discussions regarding the potential efficiency and privacy limitations of our method and suggestions on their future mitigation.

Table 1: Learning on distributed ego-networks suffers from incomplete local neighborhood information.

Avg. Accuracy	Cora	LastFM-Asia
Local	0.8006 (± 0.0028)	0.8678 (± 0.0013)
FedAvg	0.7982 (± 0.0012)	0.8582 (± 0.0006)
FedProx	0.7991 (± 0.0013)	0.8594 (± 0.0027)
Global	0.8399 (± 0.0081)	0.8792 (± 0.0038)

Our work contributes to both problem innovation and technique designs:

- We study the novel problem setting of FGL over distributed ego-networks and its essential challenge, which motivates a novel FGL framework based on node embedding sharing, as orthogonal to traditional FGL through model sharing.
- We design a contrastive learning mechanism for stabilized embedding sharing and employ a secure sharing protocol to protect individual local node identities and embeddings. We also conduct detailed discussions about the potential drawbacks of our proposed methods and their future mitigation.
- We conduct comprehensive experiments and in-depth analysis on four real-world graph datasets to demonstrate the effectiveness of our proposed methods over different model-sharing mechanisms and against SOTA baselines.

2 Related Works

2.1 Node Classification

Node classification is a pivotal graph mining task that can be applied across numerous real-world scenarios for diverse applications. The traditional data mining techniques for node classification include graph kernels [39, 44, 53, 54], which measure similarity between graphs and identify the recurring patterns or motifs for classifying nodes; label propagation [52, 65] by which the label or information of nodes are propagated through graph edges iteratively till convergence for node class inference; shallow embeddings [11, 33] which incorporate random walk algorithms to generate numerical embeddings that preserve local neighborhood information of each node, and these node embeddings can be used for downstream node classification.

Recently, more advanced techniques for learning on graphs have emerged and gained extensive attention, such as the graph neural networks (GNNs) of GCN [23, 36] and graph transformers [19, 56]. GCNs can iterative update node representations by aggregating

information from nodes' neighbors through message-passing mechanisms. For a node classification task, GNNs can be trained with the supervision of labeled nodes to predict unlabeled nodes. Graph transformers adapt the transformer architecture to learn on graphs by computing the connectivity via self-attentions. It can realize a long-distance dependency within graphs therefore learning the entire graph information, which improves the effectiveness for downstream tasks including node classification.

2.2 Federated Learning with Graph Data

Federated learning (FL) is developed for realizing information sharing among distributed data. In many real-world scenarios where graphs are distributed, FL can be employed as an effective approach to allow collaboration among these graphs while preserving their data privacy, which is denoted as federated graph learning (FGL). Recently, many works have started to study FGL which integrates GNN techniques with FL algorithms for downstream graph tasks like node classification [3, 12, 45, 50], graph classification [16, 42, 49, 51], and link prediction [18, 32].

Among them, FGL for node classification is an important application and mainly lies in the setting of FL with subgraphs [1, 24, 55, 58, 60, 61]. Previous works in FGL with subgraphs mainly aim to address 1) the missing links across distributed subgraphs [24, 58, 61, 63], or 2) graph data heterogeneity and distribution shift [1, 13]. Our setting of FGL over distributed ego-networks is closer to the setting of the former due to the consideration and mitigation of incomplete local information, while orthogonal to the latter which focuses on better ways to share models. However, these subgraph FGL studies focus on the disjoint subgraphs and try to recover the missing links by learning the neighbor distributions, but these missing links do not exist in any clients' graph and no real supervision can be used to guide the learning of real missing links. In contrast, our setting studies the distributed overlapping ego-networks where the missing information of one ego-network actually exists in other ego-networks, and we study the sharing of such real information across ego-networks via embeddings. Recently, there have emerged several works studying federated embedding sharing [4, 5, 46, 59], but they are mostly designed for vertical FL settings with different types of relations on knowledge graphs and are divergent from our FGL over ego-network setting.

Additionally, several recent studies have explored settings somewhat relevant to FL with ego-networks [27, 34, 47, 48]. However, our work intrinsically distinguishes itself from these existing studies in several key aspects. While these prior works focus on applying FL to bipartite graphs for recommendations, characterized by specific graph schemas, user-item interactions, and downstream tasks, our research investigates FL on (generic) ego-networks. Another key distinguishing aspect is our aim to address the unique challenges posed by the natural incompleteness inherent in both ego node features and the features and link structures of the ego's neighboring nodes. Moreover, our work focuses on the node embedding sharing approach, as well as the interrelated privacy-preserving concerns, an aspect that has been limited explored in previous studies.

3 Problem Formulation

Distributed Ego-networks. This work studies the novel problem setting of FGL over distributed ego-networks. In this FGL system, suppose there exist a server and m clients $\mathbb{C} = \{C\}^m$, each client $C_a \in \mathbb{C}$ representing a user/individual (a.k.a., ego) who possesses its ego-network $g_a = (V_a, E_a, \mathbf{X}_a, \mathbf{y}_a, I_a)$, where V_a and E_a represent the node set and edge set, respectively, $\mathbf{X}_a$ and $\mathbf{y}_a$ are node features and labels w.r.t. V_a, and I_a is the set containing the unique identities corresponding to V_a. From the global level, $I_a \subset_\varnothing \mathbb{I}^n$, where $\mathbb{I}^n$ represents the unique identities of overall n nodes across all ego-networks $\mathcal{G} = \{g\}^m$. The existing of global identities is realistic yet common in various scenarios like a username in e-commerce or phone number in the contact network. For a client C_a, its ego-network g_a will always overlap with one or more ego-networks in other clients, such that g_a and its overlapped ego-network g_b have

$$g_a \cap g_b \neq \varnothing : V_a \cap V_b = \{v : v \in V_a \text{ and } v \in V_b\}.$$

For a node $v \in V_a \cap V_b$, its neighbors $\mathcal{N}_a(v)$ in g_a and neighbors $\mathcal{N}_b(v)$ in g_b have

$$\mathcal{N}_a(v) \cap \mathcal{N}_b(v) \subseteq_\varnothing \mathcal{N}_a(v) \text{ or } \mathcal{N}_b(v).$$

In practice, $\mathcal{N}_a(v)$ and $\mathcal{N}_b(v)$ can hardly be equal.

Node Classification. Node classification is an important graph learning task in the scenarios of FGL over distributed ego-networks. In this work, we focus on federated node classification across ego-networks. We employ GNNs for learning on local ego-networks $\mathcal{G}$, which in general can be formulated as:

$$h_v^{(l+1)} = \sigma\left(\left[\mathbf{W}^{(l)} \operatorname{agg}\left(\{h_u^{(l)}; u \in \mathcal{N}(v)\}\right), \mathbf{B}^{(l)} h_v^{(l)}\right]\right), \forall l \in [0, L), \tag{1}$$

where $h_v^{(l+1)}$ represents the node embeddings of v at the $(l+1)^{\text{th}}$ layer out of the total L layer, $\sigma(\cdot)$ and $\operatorname{agg}(\cdot)$ refer to activation function and aggregation function, respectively, and $\Theta_{\text{gnn}} = [\mathbf{W}, \mathbf{B}]$ are parameters of GNNs. The node embeddings $h_v^{(l+1)}$ are computed by aggregating its own information (embeddings) and the information $h_u^{(l)}$ propagated from its neighbors $u \in \mathcal{N}(v)$ from the previous layer l. On top of the GNNs, we adopt linear layers as node classifiers $\Theta_{\text{clf}} = [\mathbf{w}, b]$, which take node embeddings generated from GNNs and predict the class of nodes, i.e.,

$$\hat{y}_v = \mathbf{w}^{\mathrm{T}} h_v + b. \tag{2}$$

The local empirical risk $\mathcal{R}_a$ is the cross-entropy loss for node classification:

$$\mathcal{R}_a(\Theta_a; g_a) \coloneqq \mathcal{L}_{\text{nc}} = -\sum_v y_v \log(\hat{y}_v), \tag{3}$$

where $\Theta_a = [\Theta_{\text{gnn}_a}, \Theta_{\text{clf}_a}]$. For each client C_a, the local model Θ_a is trained with a training node set V_a^{tr} by optimizing the local risk,

$$\Theta_a^* = \arg\min \mathcal{R}_a(\Theta_a; g_a). \tag{4}$$

Federated Learning. Federated learning (FL) involves information sharing which is usually realized by sharing local models. Regarding the model sharing-based FL methods, the goal is to optimize models by minimizing the global empirical risk $\mathcal{R}$:

$$\mathcal{R}(\Theta; \mathcal{G}) \coloneqq \mathop{\mathbb{E}}_{a \in [1,m]} [\mathcal{R}_a(\Theta_a; g_a)]. \tag{5}$$

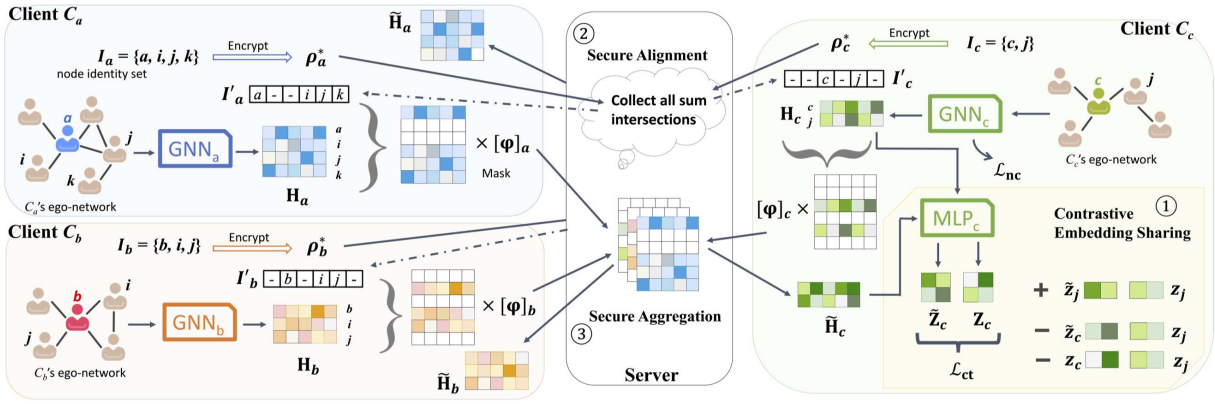


Figure 2: Overview of FedSCem. Each client possesses a node identity set I and securely communicates with the server to obtain the aligned view of the node identities I' (② secure alignment)[1]. Based on I', the local embeddings $\mathbf{H}$ are rearranged and encrypted for secure aggregation (③). After receiving the global embeddings, clients perform contrastive learning for local and global embedding fusion ①.

4 Methods

FGL over distributed ego-networks confronts the unique challenge of natural incompleteness of local graphs which impedes the model sharing-based FGL methods. To address it, we propose a novel approach that aims to share actual graph information across clients by sharing the node embeddings of local ego-networks (Section 4.1). To technically realize the embedding sharing, we introduce a contrastive embedding sharing method that guides the local models by contrastive learning between local embeddings and global embeddings (Section 4.2). Moreover, regarding the privacy concerns of sharing embeddings, we propose a secure embedding sharing protocol comprising secure alignment and secure aggregation that can protect the local node identities and local embeddings (Section 4.3). Additionally, despite the less focus on the efficiency improvement of our proposed methods, we theoretically and empirically analyze the efficiency of our methods, discuss the potential efficiency and privacy risks, and propose future directions for improvements (Section 4.4).

4.1 Embedding Sharing for Federated Node Classification across Ego-networks

The unique challenge of incomplete local neighborhood information in FGL with distributed ego-networks necessitates the sharing of actual graph information, which can help complete the missing neighborhood information across clients that simply sharing models cannot capture. We note that a recent work FedSage+ [61] studies the problem of missing links across disjoint distributed subgraphs in the cross-silo FGL setting. It incorporates a missing neighbor generator to simulate the missing links across disjoint subgraphs from the local distributions. The original intention of the missing neighbor generator is to recover those cross-subgraph missing links; however, the real cross-subgraph links that should be utilized to supervise the process of link generation do not exist in FedSage+'s setting. Therefore, the missing neighbor generator can only generate virtual neighbors that might be similar to the learned node feature distributions from other clients, which can be wildly inaccurate and lead to noisy neighborhoods. In contrast, in this work, we aim to retrieve information about the real neighbors across clients and share the actual local graph information by sharing node embeddings. Through this approach, local clients can retain accurate distributions of those neighbors that are missing in their local ego-networks.

For sharing embeddings, clients first obtain their node embeddings $\mathbf{H}_a$ by passing the original node features $\mathbf{X}_a$ into GNNs, as in Equation 1, and then upload these node embeddings to the server. The server receives the shared local node embeddings and aggregates them to compute the global node embeddings:

$$\tilde{\mathbf{H}} = \underset{a\in[1,m]}{\text{agg}}\ (\{\mathbf{H}_a\}), \tag{6}$$

where aggregation function $\text{agg}(\cdot)$ can be adapted to various algorithms. We employ averaging for aggregation in this work because it is basic and intuitive. As each local client possesses a different set of nodes V_a, we compute the average global embeddings of a node v if it appears in multiple local ego-networks, i.e.:

$$\tilde{h}_v = \underset{\Psi}{\mathbb{E}}(h_{a_v}),\ \text{where}\ \Psi = \{a \in [1,m] \,\big|\, v \in V_a\} \tag{7}$$

The aggregated global node embeddings $\tilde{\mathbf{H}}$ will then be sent back to local clients.

Embedding Sharing v.s. Model Sharing. The approach of embedding sharing in FGL diverges from traditional model sharing techniques, which focus on disparate problems and serve to tackle generic challenges for FGL. For instance, in the FGL setting where local clients contend with inadequate data for training unbiased local models yet target congruent task objectives, model sharing can confer advantages to the local clients. Conversely, when local clients possess adequate data and target incongruent task objectives, the utility of model sharing becomes trivial and even detrimental. The model sharing methods are specifically devised to study these problems of trading off between personalization (i.e., self-training) and sharing (i.e., FL) which arise from distinct scenarios characterized by varying datasets and tasks. Unlike the model sharing methods, our embedding sharing method tackles the distinctive obstacle of absent neighborhood information within ego-networks in the FGL setting, which is orthogonal to the model sharing methods. Hence,

[1]The aligned views of node identities I' shown in the figure do not represent the actual alignment order. Instead, we provide a simplified demonstration of I' for clarity. For a detailed explanation of the exact alignment process, please refer to Section 4.3.1.

the embedding sharing method can be integrated with other model sharing methods to further enhance utility, particularly in the FGL scenarios where local graph information remains incomplete.

4.2 Contrastive Learning for Local and Global Embedding Fusion

The embedding sharing procedure can be analogous to the basic FedAvg [31] method, as described in Section 4.1. However, after local clients receive the aggregated global embeddings, *how should the local clients adapt the global embeddings into their local model training?* Straightforwardly, it seems plausible that local clients substitute their original embeddings with global embeddings, akin to how clients update local models in FedAvg. However, this way is in fact infeasible due to the gap between global embeddings and local embeddings and models, that is, unlike the models, embeddings encapsulate real graph information and cannot be seamlessly swapped out. Specifically elaborating, the local models are trained on local distributions to output the local embeddings, while the global embeddings aggregate these outputs of local models and are unsuitable as inputs for training these same models. To this end, we propose a contrastive embedding sharing method by incorporating contrastive learning [14] that can guide the local models to generate local embeddings approaching the globally aggregated embeddings for the same nodes while separating the different nodes, therefore realizing the adaption of global graph information into local models. The procedure is depicted in Figure 2 on the right.

After the server computes the aggregated global node embeddings $\tilde{\mathbf{H}}$ (to be explained in Section 4.3), client C_c will receive partial $\tilde{\mathbf{H}}$ comprising the aggregated global embeddings of those nodes in g_c that also appear in other ego-networks, i.e., $\tilde{\mathbf{H}}_c$. To contrastively learn between the local embeddings $\mathbf{H}_c$ and global embeddings $\tilde{\mathbf{H}}_c$ for making the same nodes similar while diverging different nodes, we first incorporate a multilayer perception (MLP) as a projection head $f(\cdot)$ to transform $\tilde{\mathbf{H}}_c$ and $\mathbf{H}_c$ into the corresponding metric embeddings $\tilde{\mathbf{Z}}_c$ and $\mathbf{Z}_c$, respectively:

$$\tilde{\mathbf{Z}}_c = f(\tilde{\mathbf{H}}_c; \Theta_{\text{mlp}}),\ \mathbf{Z}_c = f(\mathbf{H}_c; \Theta_{\text{mlp}}) \tag{8}$$

The nonlinear projection head $f(\cdot)$ is added on the top of a GNN to enhance the representation quality of the preceding convolutional layer through nonlinear transformations [6], and it will be discarded after training is complete. The metric embeddings $\tilde{\mathbf{Z}}_c$ and $\mathbf{Z}_c$ are then used to compute the contrastive loss $\mathcal{L}_{\text{ct}}$, for which we incorporate the commonly used NT-Xent loss [6]. For a node j in client C_c, the NT-Xent contrastive loss between its local metric embedding $z_j = f(h_j)$ and its global metric embedding $\tilde{z}_j = f(\tilde{h}_j)$ is calculated by

$$\ell(z_j, \tilde{z}_j) = -\log \frac{\exp(\text{sim}(z_j, \tilde{z}_j)/\tau)}{\sum_{k=1}^{2|V_c|} \mathbb{1}_{[k \neq j]} \exp(\text{sim}(z_j, z_k)/\tau)}, \tag{9}$$

where τ is the temperature hyperparameter, and $\text{sim}(\cdot)$ uses a cosine similarity function $\frac{z_j * z_k}{\|z_j\| \|z_k\|}$. $\mathbb{1}_{[k \neq j]} \in \{0, 1\}$ is an indicator function that only returns 1 when $k \neq j$. Hereby, the contrastive loss for the positive pair $(z_j, \tilde{z}_j)$ is computed by contrasting all negative pairs $\{(z_j, \tilde{z}_k), (z_j, z_k) \mid k \in V_c\}$ (V_c represents the node set of ego-netowrk on client C_c), aiming to maximize the agreement between the same nodes and minimize the agreement between different nodes, as depicted by ① in Figure 2. Overall, the contrastive loss $\mathcal{L}_{\text{ct}}$ for client C_c is

$$\mathcal{L}_{\text{ct}} = \sum_{j \in V_c} \ell(z_j, \tilde{z}_j). \tag{10}$$

The local GNN is then updated end-to-end by the backpropagation of $\mathcal{L}_{\text{ct}}$. As this procedure is trained unsupervised and invariant to the downstream task, we integrate the node classification loss $\mathcal{L}_{\text{nc}}$ with $\mathcal{L}_{\text{ct}}$ for co-training the local models. Thus, for a client C_c, its local empirical risk is

$$\mathcal{R}_c(\Theta_c; g_c) := \mathcal{L}_{\text{ours}} = \sum_{j \in V_c} -y_j \log(\hat{y}_j) + \mu * \ell(z_j, \tilde{z}_j), \tag{11}$$

where $\hat{y}_j$ is the predicted node label, and z_j and $\tilde{z}_j$ are the metric embeddings of node j's local and global node embeddings. We introduce a hyperparameter μ to weight the contrastive loss $\mathcal{L}_{\text{ct}}$.

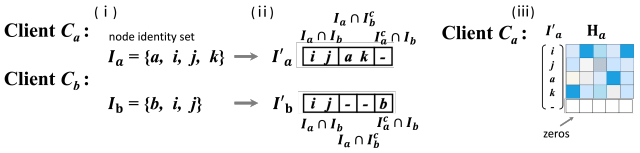


Figure 3: Securely aligning node identities.

4.3 Secure Embedding Sharing among Clients and the Server

Given that node embeddings encapsulate sensitive original graph data, sharing them directly invokes privacy concerns regarding potential local data leakage. Moreover, considering that clients' ego-networks vary in size and node sets, the information regarding how many and which nodes a client possesses is also private and requires privacy protection.

As discussed in Section 4.1, the procedure of embedding sharing among clients involves: 1) the server aligns the overlapped local node embeddings across the clients according to original node identities, 2) the server aggregates these local embeddings based on alignment and computes the global node embeddings, and 3) the server sends back the globally aggregated node embeddings to clients. The potential privacy concerns arise from: 1) the number of nodes and the node identities within the clients being exposed to the server, 2) the local data information being leaked through actual embeddings to the server, 3) the client's local data information being leaked through aggregated global embeddings to other clients. To address these privacy concerns, we propose a secure embedding sharing protocol that can prevent these information leakages among clients and the server. Specifically, we utilize a secure alignment method to ensure that the server does not learn the node identity belongings of clients and a secure aggregation method by which the shared local embeddings are encrypted for aggregation in the server. Regarding the privacy concern in downloading the globally aggregated embeddings, differential privacy techniques can be adapted, which we discuss in Section 4.4.2.

4.3.1 ***Secure Alignment of Node Identities.*** Motivated by [21] designed for computational phenotyping that aims to align clinical concepts (phenotypes) before aggregating model gradients, we

adapt the idea of the secure alignment of phenotypes into our node embedding sharing setting to realize the secure alignment of node identities before embedding aggregation, which can protect the size of node sets and the node identities within each client from the server.

As illustrated by (i) in Figure 3, clients differ in node sets V leading to non-aligned node identity sets I. To prevent the exposure of local node identities to the server, each client C_a first encrypts the node identity set I_a to a polynomial (function) $\rho_a(\gamma) = (\gamma - \gamma_{a_1}) \cdots (\gamma - \gamma_{a_{|I_a|}})$, where $\gamma_{a_i} \in I_a$ and a node identity γ belongs to I_a if and only if $\rho_a(\gamma) = 0$. To avoid factorization of the polynomials for preventing the server from inspecting the identities, clients compute the polynomial ρ_a^* by multiplying ρ_a with $(\gamma - \xi)$ using a prime random number ξ and compute its modulus (%) using a prime random number β,

$$\rho_a^*(\gamma) = [\rho_a(\gamma) * (\gamma - \xi)] \ \% \ \beta, \text{ where } \xi \notin I_a. \tag{12}$$

Then a node identity γ belongs to I_a if and only if $\rho_a^*(\gamma) = 0$. The server receives the uploaded polynomials $\{\rho^*\}^m$ from m clients and finds the pair-wise intersection $I_a \cap I_b$ between two clients C_a and C_b by computing the sum of their polynomials $[\rho_a^*(\gamma) + \rho_b^*(\gamma)] \ \% \ \beta$, from which a node identity $\gamma \in I_a \cap I_b$ if $[\rho_a^*(\gamma) + \rho_b^*(\gamma)] \ \% \ \beta = 0$. The server then sends back $(m-1)$ polynomial to each client, and the client can check whether each of its node identity γ is in the intersections with others' node identity sets. Referring to the protocol in [21], the server collects the sizes of all combinations of intersections and has all clients agree on the same order of all intersections (with a total length of n), and the size of every combination of intersections is allocated. Herewith, clients can align their nodes according to the agreed global ordering. For example, in (ii) of Figure 3, considering two clients C_a and C_b, given the global ordering $I_a \cap I_b, I_a \cap I_b^c, I_a^c \cap I_b$, and the size of each intersection, clients can rearrange these nodes correspondingly to obtain the aligned view I' of I (nodes within the sections are sorted). The alignment is secure because clients are ignorant of the elements of others' I', and the server cannot inspect the elements of all intersections. With the alignment, each client uploads the transformed version of local node embeddings based on I' by filling empties with $\mathbf{0}$ (see (iii) in Figure 3). Since the secure alignment of node identities is computed only once with implementations like utilizing modulus and sparse tensors, it will not lead to extra burdens on time and memory efficiency.

4.3.2 ***Secure Aggregation for Local Embeddings.*** After secure alignment, the clients need to upload the local node embeddings based on the aligned order for aggregation. The local node embeddings are sensitive information because they encode not only the nodes' features but also the feature and link information of their neighbors. To prevent the potential exposure risk of local data information to the server via local embeddings, we incorporate a secure aggregation protocol. An existing work [2] improves previous secure aggregation protocols by reducing the transmitted data (w.r.t. efficiency) and resolving the privacy risks of clients' dropout. Motivated by it, we design a secure aggregation protocol to securely obtain the averaging aggregation of the local node embeddings.

In this protocol, we have clients agree on the pair-wise Diffie-Hellman key agreement [7] for encrypting local embeddings, in which clients share a public generator φ and a prime modulus β while each client C_a keeps its own secret key κ_a. The public generator φ (% β) pairs with C_a's secret key to generate the public key φ^{κ_a} for client C_a, which is safe to reveal. The server then collects all m public keys from m clients and sends all other $(m-1)$ public keys to each client. Client C_a raises the received public key φ^{κ_b} from client C_b to the power of its secret key κ_a to get the shared secret $\varphi^{\kappa_b \kappa_a}$ with client C_b. These pair-wise shared secret keys applied to local embeddings will then be canceled out during the sum aggregation.

As the local node embeddings are transformed and uploaded after secure alignment whose non-overlapping nodes' positions are filled with zeros, simply averaging them can yield incorrect global embedding for these non-overlapping nodes. To address it, we employ an additional activation mask $\mathbf{o}_a = \{0,1\}^n$ for each I'_a of client C_a which indicates whether a position in I'_a has node identity. We also apply secure aggregation for the activation masks $\{\mathbf{o}\}^m$ to ensure the security of node identities and their alignments. Finally, the averaged global embeddings are computed by dividing the sum aggregation of local embeddings by the sum aggregation of activation masks.

Regarding the privacy risks of client dropout in secure aggregation, the idea of double-masking in [2] can be integrated into our protocol. Besides, the potential privacy concerns can also arise from the aggregated global embeddings. For these privacy concerns, we provide more discussion in Section 4.4.2.

4.4 Discussions on Efficiency and Privacy

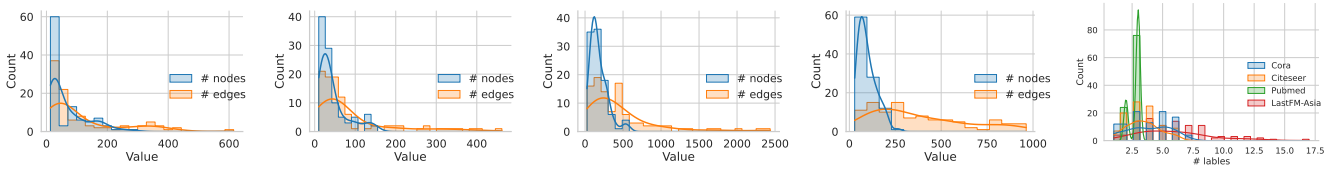
4.4.1 ***Efficiency Analysis.*** This work does not focus on efficiency improvement. The main efficiency concerns can arise from sharing node embeddings with a large number of nodes. Here, we analyze the potential efficiency burdens that can be brought by node sharing with contrastive learning and secure protocol, as well as the potential future exploration to mitigate the efficiency issues.

Extra Costs of Contrastive Embedding Sharing. Compared to the model sharing FL methods like FedAvg, our contrastive learning-based embedding sharing method only increases the computation cost from the projection head, which can be a one-layer MLP with small hidden states and will not cause extra computation burden. Regarding communication and memory cost, as our method only transmits node embeddings, it relieves the cost of model transmission and the memory cost of model storage in the server, but it can encounter overwhelming communication costs with a large number of total nodes n, which could be $O(m * n)$ for m clients at the worst case.

Extra Costs of Secure Embedding Sharing. With the secure alignment protocol, it can incur extra $O(m)$ computation cost for all clients to compute intersections and $O(m * n)$ memory cost for all clients to store the aligned view of node identity sets. The extra communication cost is $O(m^2)$ for transmitting all intersections between clients and the server. However, the aligning process is computed once ahead, which will not cause strong efficiency drawbacks. For the secure aggregation, each client costs $O(1)$ for key agreements and encrypting local embeddings and activation masks which leads to an extra $O(m)$ computation cost, and the

Table 2: Statistics of ego-networks from four datasets.

Dataset	♯ total nodes nodes	♯ total edges edges	♯ features	♯ classes	density	100 ego-networks avg. ♯ nodes	avg. ♯ edges	avg. ♯ classes	avg. density
Cora	1,523	3,006	1,433	7	1.97	66.88	117.51	3.86	1.76
Citeseer	1,081	2,192	3,703	6	2.03	43.68	87.76	3.40	2.01
Pubmed	6,977	19,129	500	3	2.74	176.93	451.58	2.76	2.55
LastFM-Asia	3,671	15,083	7,842	18	4.11	96.45	375.05	5.96	3.89



(a) Cora (b) Citeseer (c) Pubmed (d) LastFM-Asia (e) Label distirbution

Figure 4: Distributions of ego-networks w.r.t. the numbers of nodes, edges, and labels on all datasets.

additional activation masks lead to extra $O(m * n)$ memory and communication costs. Similarly, the key agreements and activation masks are computed once and will not impose strong computation burdens.

Potential Mitigation. As the main efficiency concerns arise from large numbers of clients and nodes, apart from the generic improvement regarding communication cost like sampling participating clients and reducing the frequency of sharing, more dedicated techniques can be designed w.r.t. a large number of nodes such as prototyping nodes and only sharing the local embedding of prototypes, which can reduce the communication and memory burdens. Furthermore, from the perspective of FL, clustering clients and conducting within-cluster sharing can also be a possible solution to improve efficiency.

4.4.2 ***Privacy Analysis.*** Although our secure embedding sharing protocol protects the node identities and actual local embeddings for aggregation, the global embeddings broadcast to local clients can be vulnerable to inference attacks. Additionally, our method does not have a specific privacy protocol designed for local models because no model is shared if our method is used directly based on local self-trained models. However, when integrating our method with other FL methods based on model sharing, the aggregated global model can be exposed to attacks. Furthermore, our problem setting studies FGL over distributed ego-networks, which often follows the cross-device setting where the number of clients is large and certain client sampling techniques are usually employed or clients can fall off-line. This can raise privacy concerns during secure aggregation regarding clients dropping out, i.e., the paired shared secret masked on a sampled client' local embeddings cannot be canceled out during aggregation because of the absence of the client sharing the paired secret with the sampled client.

Potential Mitigation. Regarding the risk of broadcasting global embeddings, a potential approach to address it can be incorporating differential privacy techniques [8, 15, 62] that add noise to the local embeddings therefore protecting the embeddings from inference attacks, which can be easily adapted into our method to prevent the potential data leakage from the aggregated global embeddings. To address the privacy risk from unprotected models when combining our methods with model sharing, a possible solution is to use a secure aggregation protocol to encrypt the shared local models/gradients and use differential privacy to protect the downloaded aggregated models/gradients. For the issue of client dropout, the double-masking techniques in [2] which generate additional individual keys for clients and rely on Shamir's t-out-of-n [38] for secret sharing can be engaged, so that the server can reconstruct the shared or individual keys and starts to cancel out the masks even when some of the clients are dropout. Furthermore, as the secure aggregation focuses only on protecting the clients in a single round, more advanced techniques like [40] can be leveraged to further secure the secure aggregation for multiple rounds of FL.

5 Experiments

5.1 Experimental Settings

5.1.1 ***Data.*** To experimentally demonstrate our methods in the unique yet novel setting of FGL over distributed ego-networks, we randomly sample 100 2-hop ego-networks for each of the four public benchmarking datasets, including three publication networks Cora [30], Citeseer [10], Pubmed [37], and a social network LastFM-Asia [35]. We employ these public datasets because of the lack of access to real social networks (most open-source social networks are preprocessed w.r.t. the consideration of privacy and lead to distorted information like fake density, adding noise, etc.), as well as the three publication datasets are used by most node classification works and are also used for simulating social networks. Additionally, since our setting only requires ego-networks from an entire graph to be distributed to local clients, given a certain number of clients/egos, the size of an entire graph does not necessarily affect the distributions of ego-networks. The data statistics are presented in Table 2. The first five columns provide an overview of the entire dataset statistics, while the next five columns present the average statistics for 100 ego-networks. Additionally, we further analyze the data distribution for the sample ego-networks from all datasets. Figure 4 displays the distributions of numbers of nodes, edges, and labels for ego-networks. From Figure 4a-4d, most ego-networks in the four datasets contain less than 200 nodes. Additionally, Figure 4e indicates the heterogeneity w.r.t. labels where some clients

can possess only a part of the total labels, which is realistic and we intentionally retrain in our setting.

5.1.2 ***Compared Baselines.*** We compare our methods with five baselines including: Self-training (i.e., "Local" in Table 1) in which local models are trained standalone, FedAvg [31] in which local models are aggregated by averaging, FedProx [25] which advances FedAvg by adding a proximal term to the local training loss for addressing data heterogeneity issues, LocalSage+ [61] in which the local GNN models are trained together with local NeighGen for generating "virtual" links, and FedSage+ [61] in which the local GNN models and NeighGen models are federated trained. We do not include global training as a baseline due to the lack of a global model in our setting.

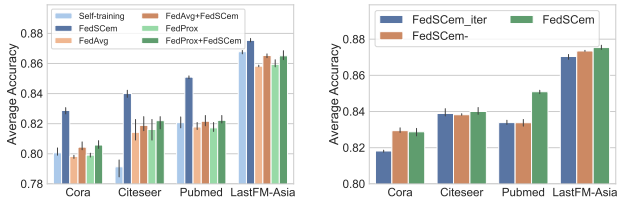
5.1.3 ***Our Models.*** We denote our model as FedSCem which co-trains the contrastive learning with node classification. Besides, we also experiment with two variants including FedSCem $_{iter}$ which trains node classification and contrastive learning iteratively by batch, and FedSCem- which removes the local training step from FedSCem that is trained before the cotraining within a communication round. Additionally, we adapt FedSCem to model sharing based FL methods, i.e., FedAvg+FedSCem and FedProx+FedSCem.

5.1.4 ***Hyper-parameter settings.*** For all the methods, we utilize a 2-layer GCN [23] model and a linear layer as a node classifier, both with the hidden size of 64. For the projection head for contrastive learning, we use a 1-layer MLP with the output dimension of 32. We incorporate Adam [22] optimizer for model updating with the learning rate of 0.0005 and the weight decay of $5e-5$. The local epoch for FL is set to 1, and the hyperparameter weighting the proximal term in FedProx is set to 0.01. For our methods, we tune μ and τ within $[0.5, 1, 5, 10]$ and $[0.01, 0.1, 1, 5, 10]$, respectively, and set $(\mu = 5, \tau = 10)$ for Cora, $(\mu = 10, \tau = 1)$ for Pubmed, and $(\mu = 1, \tau = 0.1)$ for Citeseer and LastFM-Asia datasets. We run all experiments for 3 repeats with different seeds on a server with eight 48GB NVIDIA Quadro RTX 8000 GPUs. All codes and data are provided in this repository[2].

5.2 Performance Analysis

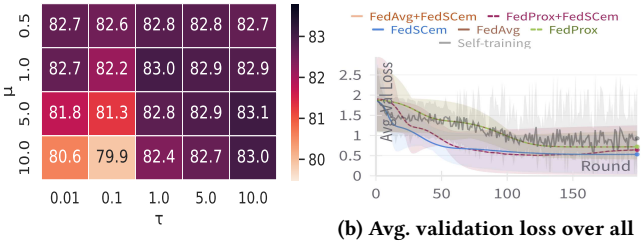
5.2.1 ***Overall Performance.*** Table 3 displays the comprehensive results of utilities (average accuracy) and efficiency (elapsed time per round and GPU memory allocated) evaluating all methods on four datasets each with 100 ego-networks. It is obvious that FedSCem and its variants (FedSCem $_{iter}$ and FedSCem-) show promising improvement from self-training and outperform FedAvg and FedProx on all datasets. FedAvg+FedSCem and FedProx+FedSCem perform worse than FedSCem because 1) model sharing can fail when local information is incomplete and 2) the gap exists between embedding sharing and model sharing and sharing both can lead the performance to optimize towards the betweenness of them. LocalSage+ and FedSage+ are originally designed for cross-silo disjointed subgraphs (where the subgraphs are relatively large) and applying them directly to the distributed ego-networks is deteriorative, due to the lack of enough data at the local ego-networks

[2]https://github.com/Oxfordblue7/FedSCem.



(a) The effect of FedSCem. **(b) Ablation Study on FedSCem.**

Figure 5: Ablation studies.



(a) Effects of μ and τ on Cora. **(b) Avg. validation loss over all datasets during training**

Figure 6: Hyper-parameter and convergence analysis.

for training a good neighbor generator (it shows reasonable performance on Pubmed with relatively large ego-networks). We provide more detailed analyses below.

5.2.2 ***Ablation Study.*** The paired comparisons of FedSCem with the corresponding baselines are in Figure 5a. For all datasets, adding FedSCem to a baseline apparently improves the performance, demonstrating the effects of FedSCem on both standalone self-training and model sharing based FL methods. Although the optimization of model sharing may diverge from that of embedding sharing, combining both methods can reach an agreement between their optimization trajectories therefore still surpassing the baselines.

We also conduct an ablation study for FedSCem and its variants FedSCem $_{iter}$ and FedSCem-. FedSCem $_{iter}$ iteratively train models for the downstream task and the contrastive learning, while FedSCem- and FedSCem co-train them and FedSCem includes one more local training epoch before co-training within a communication round. From Figure 5b, in general, FedSCem performs better than FedSCem $_{iter}$ and FedSCem-.

5.2.3 ***Hyper-parameter Analysis.*** FedSCem introduces two more hyperparameters τ and μ from contrastive loss and co-training with contrastive learning, respectively. The study of τ can be found in [6]. Here, we study the effects of combinations of μ and τ on FedSCem varying within $[0.5, 10]$ and $[0.01, 10]$, respectively (see Figure 6a). It indicates that the performance is minimally affected by the varying hyperparameters with a relatively large range.

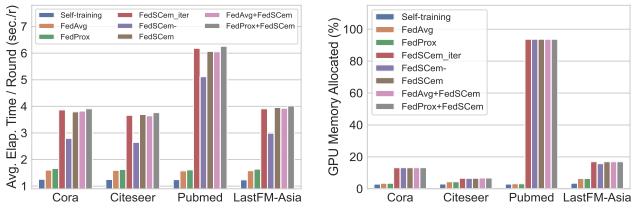
5.3 Computational Efficiency Analysis

In addition to the theoretical analysis of the efficiency of our proposed methods in Section 4.4.1, we analyze the real time and memory costs empirically. Overall, from Figure 7, our methods increase the elapsed time per round and GPU memory allocated for processes compared with baselines (LocalSage+ and FedSage+ encounter Out-of-Memory issues due to a large number of clients and are run

Table 3: Accuracy on 100 2-hop distributed ego-networks per dataset. Bold represents the best results while underline represents the second. The metric elap./r evaluates the process averaging run time per round (sec./round), and the metric mem. records the process GPU memory allocated (%).

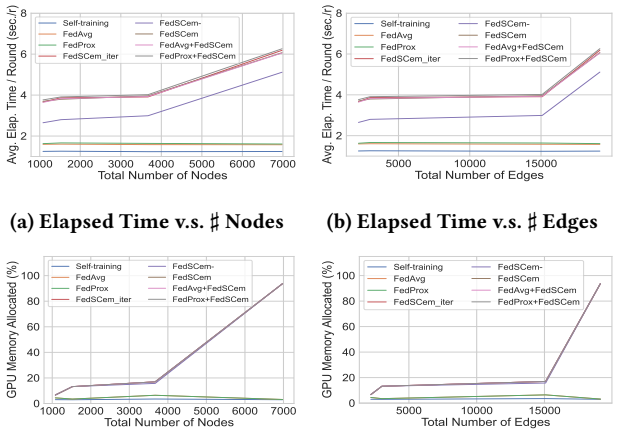
Dataset	Cora			Citeseer			Pubmed			LastFM-Asia		
Metric	avg. acc.	elap./r	mem.	avg. acc.	elap./r	mem.	avg. acc.	elap./r	mem.	avg. acc.	elap./r	mem.
Self-training	0.8006 (± 0.0028)	1.260	2.95	0.7913 (± 0.0060)	1.251	3.02	0.8207 (± 0.0037)	1.245	2.97	0.8678 (± 0.0013)	1.236	3.53
FedAvg	0.7982 (± 0.0012)	1.605	3.50	0.8142 (± 0.0075)	1.596	4.42	0.8178 (± 0.0026)	1.575	3.16	0.8582 (± 0.0006)	1.581	6.45
FedProx	0.7991 (± 0.0013)	1.665	3.50	0.8161 (± 0.0069)	1.629	4.42	0.8172 (± 0.0032)	1.611	3.16	0.8594 (± 0.0027)	1.641	6.45
LocalSage+	0.7234 (± 0.1531)	0.601*	94.54*	0.8260 (± 0.0056)	2.183	94.54	0.8189 (± 0.0041)	0.601	94.54	0.8559 (± 0.0055)	5.990	95.06
FedSage+	0.6752 (± 0.0670)	-	OOM	0.7128 (± 0.0155)	-	OOM	0.8008 (± 0.0219)	-	OOM	0.6822 (± 0.0330)	-	OOM
FedSCem $_{iter}$	0.8182 (± 0.0006)	3.867	13.21	0.8389 (± 0.0023)	3.666	6.63	0.8356 (± 0.0012)	6.186	97.36	0.8704 (± 0.0014)	3.909	16.98
FedSCem-	**0.8294** (± 0.0014)	2.799	13.23	0.8382 (± 0.0005)	2.649	6.58	0.8338 (± 0.0019)	5.115	97.36	0.8735 (± 0.0003)	2.991	15.74
FedSCem	0.8288 (± 0.0021)	3.795	13.21	**0.8402** (± 0.0018)	3.696	6.63	**0.8509** (± 0.0010)	6.069	97.36	**0.8754** (± 0.0012)	3.960	16.98
FedAvg+FedSCem	0.8044 (± 0.0030)	3.825	13.24	0.8189 (± 0.0051)	3.645	6.77	0.8222 (± 0.0028)	6.054	97.36	0.8653 (± 0.0013)	3.930	16.98
FedProx+FedSCem	0.8058 (± 0.0026)	3.909	13.24	0.8220 (± 0.0047)	3.765	6.77	0.8216 (± 0.0035)	6.261	97.36	0.8651 (± 0.0031)	4.020	16.98

* The elap./r and mem. of LocalSage+ on all datasets are evaluated standalone using only one client due to out-of-memory (OOM) issues when running with all clients.



(a) Elapsed Time v.s. Methods **(b) Memory Alloc. v.s. Methods**

Figure 7: Efficiency analysis for all methods.



(a) Elapsed Time v.s. ♯ Nodes **(b) Elapsed Time v.s. ♯ Edges**

(c) Memory Alloc. v.s. ♯ Nodes **(d) Memory Alloc. v.s. ♯ Edges**

Figure 8: The effects of graph size on efficiency.

with CPUs, thus their GPU runtimes are not available[3]). Figure 7a shows the elapsed time per round for all methods. All the four out of five of our methods (except FedSCem-) take about two times of run time of the baselines, which is because both FedSCem $_{iter}$ and FedSCem include one extra task training within each round. However, this can also lead to faster convergence which will reduce the time cost to some extent, as demonstrated in Figure 6b. In contrast, FedSCem- does not include the extra local training and its elapsed time obviously drops. Refer to Figure 5b, FedSCem- can be comparable to FedSCem in some cases, therefore there can be an option of using FedSCem- to trade off between the time efficiency and utility. Regarding the percentages of GPU memory allocated, our methods require a similar amount of memory for the smaller datasets (Citeseer) but require much more memory when the graph size grows large, as denoted in Figure 7b.

To investigate how the graph size impacts the efficiency, we depict the correlations between elapsed time/memory allocated and the number of nodes/edges of all ego-networks. From Figure 8c and 8b, the elapsed time significantly increases when the ego-network size grows beyond a certain threshold. Similarly, when the size exceeds some value, our methods require a higher amount of GPU memory allocated, from Figure 8c and 8d. This is currently a limitation of our methods which we have discussed in details in Section 4.4.1 together with the potential solutions. Moreover, both the time and memory costs rise slowly as the graph size grows within a certain range, which further empirically demonstrates that the time and memory cost of our methods is only sub-linear to the total number of nodes.

[3]The GPU runtime of LocalSage+ is reported from running on a single client/ego-network, which cannot be fairly compared.

6 Conclusion

This work studies the novel problem setting of FGL over distributed ego-networks, and aims to resolve its unique challenge of incomplete local neighborhood information, through our proposed secure contrastive embedding sharing method. Our method can contrastively learn local and global aggregated embedding fusion therefore mitigating the gaps of incomplete local neighborhood information. Meanwhile, we propose a secure embedding sharing protocol that can protect the local node identity information and real embeddings from leaking sensitive local information. The comprehensive experiments and analysis demonstrate the effectiveness of our proposed method. Furthermore, we provide detailed discussions on the potential drawbacks of our method w.r.t. efficiency and privacy, as well as the future exploration to mitigate these drawbacks.

Acknowledgments

The work is partially supported by National Science Foundation (NSF) under CNS-2124104, CNS-2125530, and IIS-2302968, National Institute of Health (NIH) under R01LM013712 and R01ES033241, and the internal funding and GPU servers provided by the Computer Science Department of Emory University. We also thank Amazon for supporting the research through the Amazon Research Awards program.

References

[1] Jinheon Baek, Wonyong Jeong, Jiongdao Jin, Jaehong Yoon, and Sung Ju Hwang. 2023. Personalized subgraph federated learning. In *ICML*. PMLR.

[2] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*.

[3] Chaochao Chen, Jun Zhou, Longfei Zheng, Huiwen Wu, Lingjuan Lyu, Jia Wu, Bingzhe Wu, Ziqi Liu, Li Wang, and Xiaolin Zheng. 2020. Vertically federated graph neural network for privacy-preserving node classification. *IJCAI*.

[4] Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. 2021. Fede: Embedding knowledge graphs in federated setting. In *Proceedings of the 10th International Joint Conference on Knowledge Graphs*.

[5] Mingyang Chen, Wen Zhang, Zonggang Yuan, Yantao Jia, and Huajun Chen. 2022. Federated knowledge graph completion via embedding-contrastive learning. *Knowledge-Based Systems* (2022).

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*.

[7] Whitfield Diffie and Martin E Hellman. 2022. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*.

[8] Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming*.

[9] Xingbo Fu, Binchi Zhang, Yushun Dong, Chen Chen, and Jundong Li. 2022. Federated graph machine learning: A survey of concepts, techniques, and applications. *ACM SIGKDD Explorations Newsletter* 24, 2 (2022), 32–47.

[10] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. 1998. CiteSeer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*.

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*.

[12] Zeli Guan, Yawen Li, Zhe Xue, Yuxin Liu, Hongrui Gao, and Yingxia Shao. 2021. Federated graph neural network for cross-graph node classification. In *CCIS*.

[13] Jiayan Guo, Shangyang Li, and Yan Zhang. 2023. An Information Theoretic Perspective for Heterogeneous Subgraph Federated Learning. In *International Conference on Database Systems for Advanced Applications*.

[14] R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *CVPR*.

[15] Xiaolin Han, Daniele Dell'Aglio, Tobias Grubenmann, Reynold Cheng, and Abraham Bernstein. 2022. A framework for differentially-private knowledge graph embeddings. *Journal of Web Semantics* (2022).

[16] 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).

[17] Chaoyang He, Emir Ceyani, Keshav Balasubramanian, Murali Annavaram, and Salman Avestimehr. 2022. Spreadgnn: Decentralized multi-task federated learning for graph neural networks on molecular data. In *AAAI*.

[18] Ce Hu, Baisong Liu, Xueyuan Zhang, Zhiye Wang, Chennan Lin, and Linze Luo. 2022. A Federated Multi-Server Knowledge Graph Embedding Framework For Link Prediction. In *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE.

[19] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*. 2704–2710.

[20] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2021. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning* (2021).

[21] Yejin Kim, Jimeng Sun, Hwanjo Yu, and Xiaoqian Jiang. 2017. Federated tensor factorization for computational phenotyping. In *KDD*.

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

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

[24] Runze Lei, Pinghui Wang, Junzhou Zhao, Lin Lan, Jing Tao, Chao Deng, Junlan Feng, Xidian Wang, and Xiaohong Guan. 2023. Federated Learning Over Coupled Graphs. *IEEE TPDS* (2023).

[25] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *MLsys* (2020).

[26] Zheng Li, Muhammad Bilal, Xiaolong Xu, Jielin Jiang, and Yan Cui. 2022. Federated Learning-Based Cross-Enterprise Recommendation With Graph Neural Networks. *IEEE Transactions on Industrial Informatics* (2022).

[27] Rui Liu, Pengwei Xing, Zichao Deng, Anran Li, Cuntai Guan, and Han Yu. 2022. Federated graph neural networks: Overview, techniques and challenges. *arXiv preprint arXiv:2202.07256* (2022).

[28] Daniel Manu, Yi Sheng, Junhuan Yang, Jieren Deng, Tong Geng, Ang Li, Caiwen Ding, Weiwen Jiang, and Lei Yang. 2021. FL-DISCO: Federated Generative Adversarial Network for Graph-based Molecule Drug Discovery: Special Session Paper. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*.

[29] Daniel Manu, Jingjing Yao, Wuji Liu, and Xiang Sun. 2024. GraphGANFed: A Federated Generative Framework for Graph-Structured Molecules Towards Efficient Drug Discovery. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2024).

[30] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* (2000).

[31] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

[32] Hao Peng, Haoran Li, Yangqiu Song, Vincent Zheng, and Jianxin Li. 2021. Differentially private federated knowledge graphs embedding. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*.

[33] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *KDD*.

[34] Liang Qu, Ningzhi Tang, Ruiqi Zheng, Quoc Viet Hung Nguyen, Zi Huang, Yuhui Shi, and Hongzhi Yin. 2023. Semi-decentralized federated ego graph learning for recommendation. In *Proceedings of the ACM Web Conference 2023*. 339–348.

[35] Benedek Rozemberczki and Rik Sarkar. 2020. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *CIKM*.

[36] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* (2008).

[37] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* (2008).

[38] Adi Shamir. 1979. How to share a secret. *Commun. ACM* (1979).

[39] Nino Shervashidze and Karsten M. Borgwardt. 2009. Fast Subtree Kernels on Graphs. In *NIPS*.

[40] Jinhyun So, Ramy E Ali, Başak Güler, Jiantao Jiao, and A Salman Avestimehr. 2023. Securing secure aggregation: Mitigating multi-round privacy leakage in federated learning. In *AAAI*.

[41] 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).

[42] Ye Tao, Ying Li, and Zhonghai Wu. 2022. SemiGraphFL: Semi-supervised Graph Federated Learning for Graph Classification. In *International Conference on Parallel Problem Solving from Nature*.

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

[44] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *JMLR* 11 (2010), 1201–1242.

[45] Binghui Wang, Ang Li, Meng Pang, Hai Li, and Yiran Chen. 2022. Graphfl: A federated learning framework for semi-supervised node classification on graphs. In *ICDM*.

[46] Maolin Wang, Dun Zeng, Zenglin Xu, Ruocheng Guo, and Xiangyu Zhao. 2023. Federated Knowledge Graph Completion via Latent Embedding Sharing and Tensor Factorization. In *2023 IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, 1361–1366.

[47] 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).

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

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

[50] Han Xie, Li Xiong, and Carl Yang. 2023. Federated node classification over graphs with latent link-type heterogeneity. In *WWW*.

[51] Han Xie, Yi Yang, Hejie Cui, and Carl Yang. 2024. Federated Learning for Cross-Institution Brain Network Analysis. In *SPIE Medical Imaging Conference*.

[52] Tian Xie, Bin Wang, and C-C Jay Kuo. 2022. Graphhop: An enhanced label propagation method for node classification. *TNNLS* (2022).

[53] Pinar Yanardag and S.V.N. Vishwanathan. 2015. Deep Graph Kernels. In *KDD*.

[54] Carl Yang, Mengxiong Liu, Vincent W Zheng, and Jiawei Han. 2018. Node, Motif and Subgraph: Leveraging Network Functional Blocks Through Structural Convolution. In *ASONAM*.

[55] Yuhang Yao, Weizhao Jin, Srivatsan Ravi, and Carlee Joe-Wong. 2024. FedGCN: Convergence-Communication Tradeoffs in Federated Training of Graph Convolutional Networks. *NeurIPS*.

[56] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. *Advances in neural information processing systems* 32 (2019).

[57] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. 2021. A survey on federated learning. *Knowledge-Based Systems* (2021).

[58] Ke Zhang, Lichao Sun, Bolin Ding, Siu Ming Yiu, and Carl Yang. 2024. Deep Efficient Private Neighbor Generation for Subgraph Federated Learning. In *SDM*.
[59] Kai Zhang, Yu Wang, Hongyi Wang, Lifu Huang, Carl Yang, Xun Chen, and Lichao Sun. 2022. Efficient federated learning on knowledge graphs via privacy-preserving relation embedding aggregation. *EMNLP* (2022).
[60] Ke Zhang, Han Xie, Zishan Gu, Xiaoxiao Li, Lichao Sun, Siu Ming Yiu, Yuan Yao, and Carl Yang. 2022. Subgraph federated learning over heterogeneous graphs.
[61] Ke Zhang, Carl Yang, Xiaoxiao Li, Lichao Sun, and Siu Ming Yiu. 2021. Subgraph federated learning with missing neighbor generation. *NeurIPS*.
[62] Shijie Zhang, Hongzhi Yin, Tong Chen, Zi Huang, Lizhen Cui, and Xiangliang Zhang. 2021. Graph embedding for recommendation against attribute inference attacks. In *WWW*.
[63] Geng Zhao, Yi Huang, and Chi Hsu Tsai. 2022. FedGSL: Federated Graph Structure Learning for Local Subgraph Augmentation. In *IEEE Big Data*. IEEE, 818–824.
[64] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).
[65] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. 2003. Learning with local and global consistency. *NIPS* 16 (2003).
[66] Wei Zhu, Jiebo Luo, and Andrew D White. 2022. Federated learning of molecular properties with graph neural networks in a heterogeneous setting. *Patterns* 3, 6 (2022).