Received 16 December 2022; accepted 3 January 2023. Date of publication 11 January 2023; date of current version 25 January 2023. The review of this paper was arranged by Guest Editors Quan Chen, Jing Li, Tuo Shi, Tongxin Zhu, and Yuncan Zhang.

Digital Object Identifier 10.1109/OJCS.2023.3235593

# SC-FGCL: Self-Adaptive Cluster-Based Federal Graph Contrastive Learning

TINGQI WANG [1], XU ZHENG [1,2], LEI GAO[1], TIANQI WAN[4], AND LING TIAN [1,3]

[1]School of Computer Science, Engineering, University of Electronic Science, Technology of China, Chengdu 611731, China
[2]Kash Institute of Electronics and Information Industry, Kashi 844000, China
[3]Shenzhen Institute of Information Technology, Shenzhen 518000, China
[4]Beijing Institute of Aerospace Measurement, Testing Technology, Beijing 100000, China

CORRESPONDING AUTHOR: XU ZHENG. (e-mail: xzheng@uestc.edu.cn)

This work work was supported in part by the National Natural Science Foundation of China under Grants 62271125 and 62273071, in part by Sichuan Science and Technology Program under Grants 2022YFG0038 and 2021YFG0018, and in part by the Fundamental Research Funds for the Central Universities under Grants ZYGX2020ZB034 and ZYGX2021J019.

**ABSTRACT** As a self-supervised learning method, the graph contrastive learning achieve admirable performance in graph pre-training tasks, and can be fine-tuned for multiple downstream tasks such as protein structure prediction, social recommendation, *etc*. One prerequisite for graph contrastive learning is the support of huge graphs in the training procedure. However, the graph data nowadays are distributed in various devices and hold by different owners, like those smart devices in Internet of Things. Considering the non-negligible consumptions on computing, storage, communication, data privacy and other issues, these devices often prefer to keep data locally, which significantly reduces the graph contrastive learning performance. In this paper, we propose a novel federal graph contrastive learning framework. First, it is able to update node embeddings during training by means of a federation method, allowing the local GCL to acquire anchors with richer information. Second, we design a Self-adaptive Cluster-based server strategy to select the optimal embedding update scheme, which maximizes the richness of the embedding information while avoiding the interference of noise. Generally, our method can build anchors with richer information through a federated learning approach, thus alleviating the performance degradation of graph contrastive learning due to distributed storage. Extensive analysis and experimental results demonstrate the superiority of our framework.

**INDEX TERMS** Differential privacy, federated learning, graph contrastive learning.

## I. INTRODUCTION

In recent years, due to the outstanding performance of graph contrastive learning (GCL) in node classification, node clustering and graph classification, it has attracted widespread attention in tasks such as protein structure prediction [1] and computer vision [2]. As a self-supervised graph representation learning method, GCL is able to address the real-world problem of existing large amounts of unlabelled and unusable graph data [3], [4]. It learns the representation of such graph data as a result of pre-training, and only slight fine-tuning is required when applied to a specific task, thus unlocking significant time or equipment costs, and providing good transferability [5].

Specifically, to realize the pre-trained model, GCL typically begins by obtaining positive and negative samples from various views. It frequently employs data augmentation (e.g., node dropout, edge dropout, and feature mask) to obtain augmented graph from original graph, such as GraphCL [6]. And then uses the original graph as the anchors, with the data corresponding to the augmented graph being positive sample pairs and the other data being negative sample pairs. Anchors are used as the reference for training. Following that, the theory of maximizing mutual information (infoMAX) [7] is introduced to maximize the similarity of positive sample pairs on the hidden space while decreasing the similarity of negative sample pairs. According to this learning process, it is noticed that anchors have a decisive influence on the training effect, and more representative anchors are more conducive for generalized and stable representations [8], [9]. In the case of centralized training, GCL uses the original graph as a proper anchor.

However, in the real world, due to the large size of real-world graph data [10], [11], it is frequently scattered across multiple devices in more general cases like Internet of Things (IoTs) [12], [13], making it difficult to interoperate and resulting in data isolation with privacy protection in mind [14], [15], [16], [17]. It causes the absence structural and feature information in the stored graphs in each device [18], which lead to a lack of representative ability of origin graph [19]. Then corresponding anchors can lead to a significant reduction in the effectiveness of GCL.

In recent years, federated learning approaches [20] are proposed and flourished, which aims at solving the distributed learning of deep neural networks [21]. They usually aggregate the neural network parameters across clients by weighted averaging, and follow an iterative mode to train models without fusing all data to the central server. However, previous efforts in federated learning have been of limited use in solving the problem of GCL, where specific designs towards mitigating the missing of anchor information are imperative.

To solve the above problem, this paper proposes Self-adaptive Cluster-based Federal Graph Contrastive Learning (SC-FGCL), which allows local devices to federate with other clients to update the node embeddings obtained from clients' network layers, SC-FGCL provides local clients with critical clues to obtain anchors with richer information and thus obtain better GCL results. Simultaneously, the self-adjusting clustering method of server in SC-FGCL automatically explores for the best embedding update solution for each round, enabling each embedding to maximize its own information enrichment while avoiding interference from other noisy node embeddings.

In SC-FGCL, it is assumed that each device holds a graph locally, which can be thought of as a subgraph of the global graph with some node overlap between devices. Then, we propose a novel federal GCL framework that uploads local node embeddings to the server, updates each node embedding jointly on the server side, and server returns the result as a local GCL anchor. We present a corresponding paradigm for the local training, feature update, and communication, as well as analysis of communication and local storage consumption.

To further improve the performance for GCL, we also introduce a self-adaptive clustering method in SC-FGCL for selecting the best update solution for each node on server side. The method recursively selects the embedding with the highest similarity and tightness for the node to be updated, by adopting the calculation of the clustering internal evaluation effectiveness metric as the criterion. In addition, the selection of the threshold for the best clustering internal evaluation metric is dependent on the Calinski-arassment score value, and the server will automatically calculate and select the update policy with the higher value as the final update policy.

Finally, our framework ensures privacy security, while differential privacy is used to protect privacy during data transfer [22], [23], [24]. We conducted experiments on three benchmark datasets and the results show the superiority of our approach.

The main contribution of this paper includes:

1) We propose a novel federal GCL framework for obtaining anchor with richer information by jointly updating node embeddings, effectively mitigating GCL performance degradation due to data isolation.
2) To enable node embeddings to be updated with maximally rich information about themselves, while reducing the interference of noisy embeddings, we propose a self-adaptive clustering method that automatically selects the optimal update scheme for node embeddings.
3) The results of the extensive evaluation show that our framework is advanced compared to the baseline.

## II. RELATED WORK

### A. GRAPH CONTRASTIVE LEARNING

Inspired by contrastive learning (CL), GCL has received wide attention in the recent years, which is based on GNN for contrastive learning on graph data. Veličković et al. proposed DGI (Deep Graph Infomax) [25], which uses the concept of DIM (deep infoMax) [7] to optimize infoNCE loss in order to maximize mutual information between local and global node representations. Following that, Sun, et al. proposed InfoGraph as an extension to DGI [26]. In contrast to DGI, which focuses on node-level GCL, InfoGraph focuses on graph-level GCL. It maximizes mutual information between graph-level and substructural representations at different scales (nodes, edges, triangles). Furthermore, Sun, et al. proposed InfoGraph* for semi-supervised scenes to extend InfoGraph.

You, et al. proposed GraphCL [6], currently the most famous of GCL. Based on SimCLR [27], GraphCL obtains two views from original Graph by data augmentation, and also adopts DIM theory for optimization. The loss function of GraphCL follow normalized temperature -scaled cross entropy loss (NT-Xent) [28], [29], [30], NT-Xent is the commonly used GCL loss at present. GraphCL has rich data augmentation methods (node drop, edge drop, feature mask) and contrastive levels (node-level, graph-level, subgraph-level), which have had a profound impact on subsequent GCL research.

Current work in GCL focuses on two main aspects: data augmentation approaches, and contrastive approaches.

For data augmentation approaches, Zhu, et al. proposed GCA [31], which bases the probability of removing edges on node centrality in order to keep structural information more valuable to Graph. Hassani et al. proposed MVGRL in [32], which introduces a multi-view approach. Recently, there has also been some excellent work emerging on learnable data augmentation approaches [3], [33], [34].

For contrastive learning methods, Qiu, et al. proposed GCC [35] to improve the transferability in interactive contrastive manner. In [36], Xu, et al. proposed GraphLoG, which uses hierarchical prototypes to capture global semantic clusters while keeping local similarity, and employs EM for efficient learning. Furthermore, some recent research focuses on how negative examples are chosen *e.g.*, [37], [38]. Meanwhile, inspired by [39], [40], [41] focuses on the learning model without negative samples.

### B. FEDERATED LEARNING

Federated learning was developed to address machine learning performance degradation caused by data isolation issues, with the goal of achieving co-training while maintaining privacy and security. FedAvg [20], the industry and academic standard for federated learning, has been widely used to date. During the training process, factors such as network parameter weights, gradients, or loss of the local model are averaged on the server, thus eliminating the need to expose local data.

On this basis, various approaches to improving FedAvg have emerged. The main issues being researched in the field of federated learning are optimizing approach, and privacy protection.

Li, et al. proposed FedProx for federated learning algorithm optimization [42], which allows different workloads to be performed by taking into account the performance of different devices. FedMA [43] was proposed by Wang, et al., to build global models via layers by matching and averaging hidden elements with the same features.

For privacy security, FL commonly employs homomorphic encryption [44], differential privacy [45], and model aggregation mechanisms [46].

Currently, there are only [47] and [48] work on federal GCL, but they focuses on data heterogeneity and reducing structural interference to graph federated learning due to differential privacy via graph contrastive learning methods. Our work focuses on addressing the limitations of distributed storage for GCL through federated learning.

## III. PROBLEM DEFINITION

### A. SYSTEM SETTINGS

The system consists of one server and $N$ clients $C:\{C_1, C_2, \ldots, C_N\}$, Each device holds its own dataset $D_i : \{G_i\}$. where $G_i : \{V_i, E_i\}$ indicates a graph with node feature set $X_i = \{x_{v\in V_i}\}$ and edge feature set $Z_i = \{z_{e\in E_i}\}$. Accordingly, the adjacency matrix $A_i$ can be obtained from the $G_i$. We assume that nodes are partially overlapped between clients i.e. $G_1 \cup G_2 \cup \ldots \cup G_N = G, V_i \cap V_j \neq \emptyset, i \neq j$.

All devices federate with others to train their own local GCL pre-training models. The training is initiated by server. At the beginning, server distributes initialized network parameters to individual clients. During training process, clients first perform forward propagation locally and upload the results to server. Then, server aggregates results and sends them back to the corresponding clients. Finally, clients update local network parameters according to aggregated results.

**TABLE 1.** Notion

| Notion | Definition |
|---|---|
| $N$ | The number of clients. |
| $C$ | Clients set |
| $C_i \in C$ | The client $C_i$ in $C$. |
| $G:\{V,E\}$ | Graph data $G$ with the node set $V$ and the edge set $E$. |
| $\|V\|$ | The number of nodes in $V$. |
| $X=\{x_{v\in V}\}$ | The node feature set of $V$. |
| $Z=\{z_{e\in E}\}$ | The node feature set of $E$. |
| $\theta_i$ | $C_i$'s local neural network parameters. |
| $\boldsymbol{L}(\boldsymbol{\theta}_i)$ | loss of $C_i$ under $\boldsymbol{\theta}_i$. |
| $f$ | Mapping function. |
| $\triangle f$ | Sensitivity of differential privacy. |
| $M$ | Randomized mechanism. |
| $\Omega$ | Arbitrary set of outputs. |
| $Pr[\cdot]$ | Probability of occurrence of random events. |

### B. SECURITY

Federated learning necessitates the exchange of frequent messages between all devices. As in FedAvg, gradient, network weights are used as transmission parameters. In this process, malicious and semi-honest clients and server may capture changes in transmission and thus infer the original data through differential attacks [49]. To protect privacy, we use differential privacy, which makes it difficult to capture changes to only one element in local node embedding.

Differential privacy is defined as follows [22]:

Neighboring datasets are two datasets $D$ and $D'$ that differ by only one record. Function $f$ is able to map the dataset $D$ to the abstract range $R$: $f : D \to R$. The maximum difference between the mapping results is defined as the sensitivity $\triangle f$. Mechanism $M$ is a randomized algorithm that transforms the result of $f$.

*Definition 1:* $(\epsilon, \delta)$-differential privacy [50]. For any neighboring datasets of $D$ and $D'$, and for every set of output $\Omega$, randomized mechanism $M$ gives $(\epsilon, \delta)$-differential privacy, if $M$ satisfies:

$$Pr[M(D) \in \Omega] \leq exp(\epsilon) \cdot Pr\left[M\left(D'\right) \in \Omega\right] + \delta \tag{1}$$

### C. DESIGN OBJECTIVE

The purpose of all clients during training is to train a local representation of the nodes via contrastive learning. Our optimization goal is to minimize the node contrastive loss on all clients:

$$\underset{\boldsymbol{\theta}_1,\boldsymbol{\theta}_2,\ldots,\boldsymbol{\theta}_N}{\arg\min} \sum_{i=1}^{N} \frac{|V_i|}{|V|} \boldsymbol{L}(\boldsymbol{\theta}_i)$$

where $|V_i|$ denotes the number of $G_i$ nodes, $|V| = \sum |V_i|$, and denotes the value of the loss function of $C_i$ under $C_i$'s local neural network parameters $\boldsymbol{\theta}_i$.

The notations mentioned above as shown in Table 1.

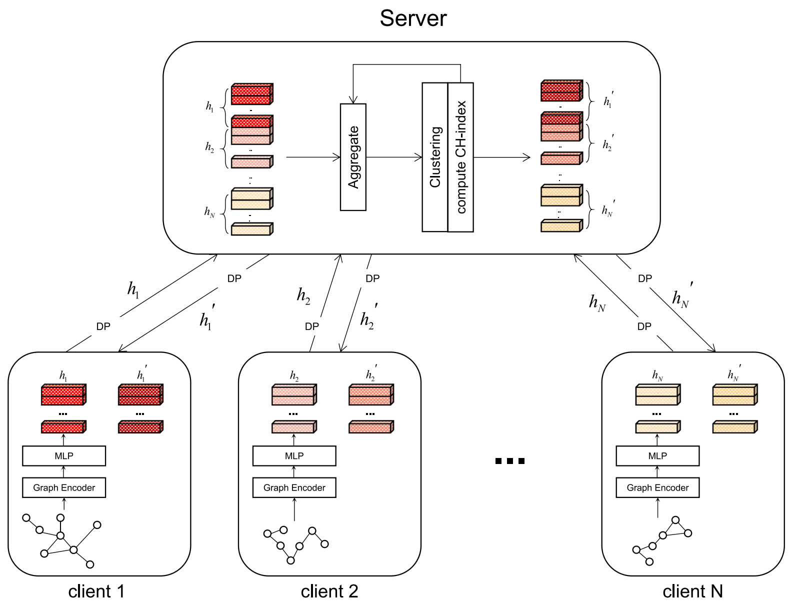


**FIGURE 1.** Overall framework of SC-FGCL.

## IV. FRAMEWORK

In this section, we first briefly outline the general training process. Next, we detail the client local training method and the server aggregation strategy. Finally, the framework's communication mechanism is defined and analyzed.

### A. OVERVIEW OF FEDERAL GRAPH CONTRASTIVE LEARNING

Motivated by the performance degradation of GCL in the distributed storage, this framework proposes that multiple clients jointly update local embeddings and use the updated embeddings as anchor for local GCL.

The overall process of the framework is shown in Fig. 1. In each training round, the client $C_i \in C$ forward propagates its graph data to get the local embedding $h_i$. Then, clients send the local embedding to server, which gets $h_S : \{h_1, h_2, \ldots, h_N\}$. On the server, the $h_S$ is updated by the self-adaptive aggregation method to get $h'_S : \{h_1,' h_2,' \ldots, h'_N\}$. After that, server sends the updated embedding back to the corresponding client. Finally, client $C_i$ receives the updated embedding $h'_i$, uses it as an anchor for contrastive learning, computes the local loss. Training stops until local models converge.

At the start of training, the server delivers the same initial graph encoder and projection head network parameters to all clients. In addition, the transmitted data between clients and server with differential privacy.

### B. CLIENT GRAPH CONTRASTIVE LEARNING MODEL

As shown in Fig. 2, in $k-th$ round training for the local client $C_i$, $G_i$ first propagates forward to obtain the node representation in the Graph:

$$h^k_{i,j} = g\left(f\left(x_{i,j}\right)\right)$$

where $f(\cdot)$ is the GCN, $g(\cdot)$ is the projection head function, and $h^k_{i,j}$ is the original embedding representation of node j in client $C_i$ at the $k-th$ round of training. Then the node embedding sequence $h^k_i$ in client $C_i$ is:

$$h^k_i = \left\{h^k_{i,0}, h^k_{i,1}, \ldots, h^k_{i,j}, \ldots, h^k_{i,|V_i|}\right\}$$

After uploading $h^k_i$ to server, $C_i$ gets the updated embedding $h^{k\prime}_i$ from the server:

$$h^{k\prime}_i = \left\{h^k_{i,0}{}', h^k_{i,1}{}', \ldots, h^k_{i,j}{}', \ldots, h^k_{i,|V_i|}{}'\right\}$$

where $h^k_{i,j}$ and $h^k_{i,j}{}'$ are positive sample pair, $h^k_{i,j}$ and the other nodes in the $h^{k\prime}_i$ are negative sample pairs.

The loss function follows the normalized temperature-scaled cross entropy loss mentioned in Section 2.1, and the loss function for node $v_{i,j}$ in the $k-th$ round,is defined as:

$$l_{i,j} = -\log \frac{\exp\left(\text{sim}\left(h^k_{i.j}, h^k_{i,j}\right)/\tau\right)}{\sum_{j'=0,j'\neq j}^{|V_i|} \exp\left(\text{sim}\left(h^k_{i,j}, h^k_{i,j'}\right)/\tau\right)}$$

where $sim(\cdot)$ is defined as:

$$sim\left(h^k_{i,n}, h^k_{i,m}\right) = \frac{h^k_{i,n}{}^T h^k_{i,m}}{\left\|h^k_{i,n}\right\| h^k_{i,m}\|}$$

The loss function of $C_i$ in the $k-th$ round is:

$$L_i = \sum_{m=1}^{|V_i|} l_m$$

Unlike the usual GCL method which uses the original graph as the anchor, our framework uses the updated embeddings returned by the server as the anchor. Meanwhile, we used the

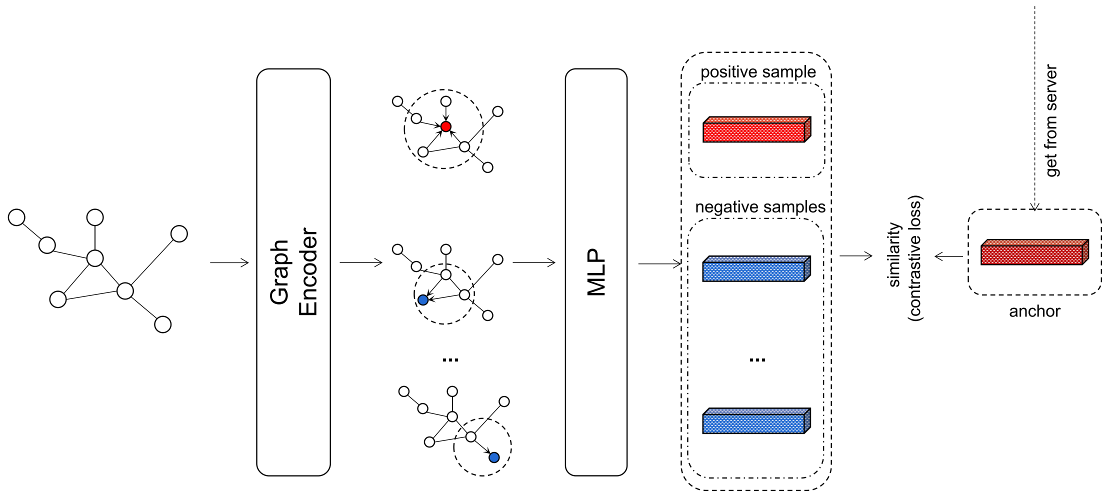


FIGURE 2. Client graph contrastive learning model.

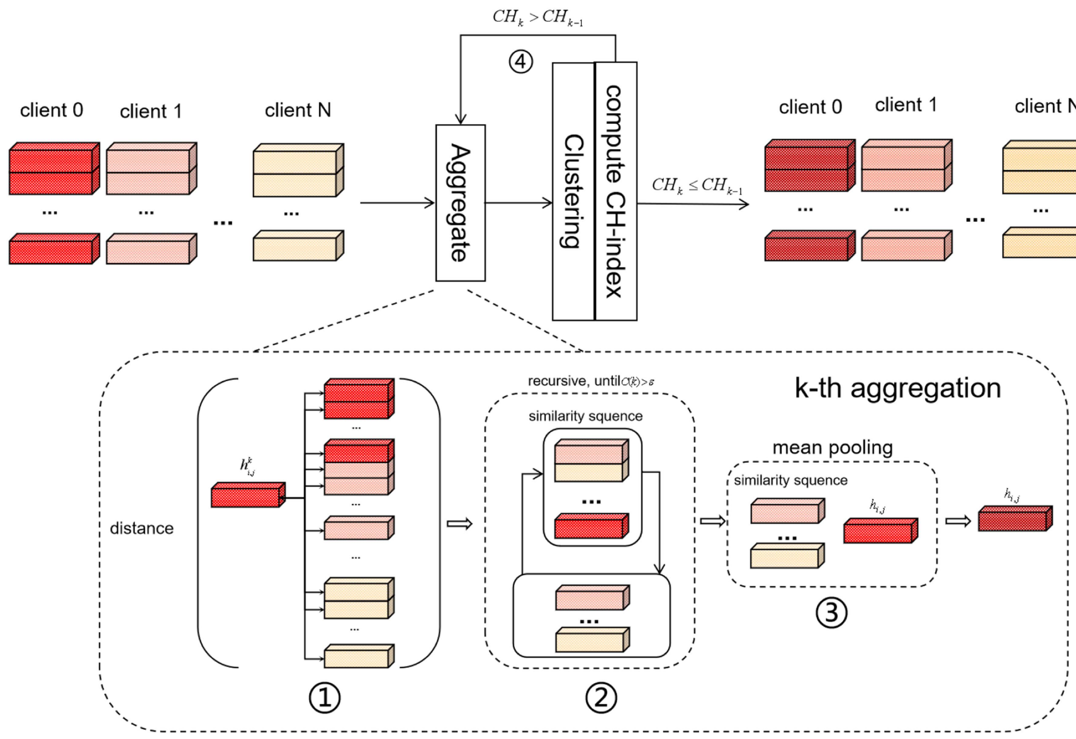


FIGURE 3. Server aggregation model.

original graph instead of data augmentation as positive and negative samples.

Our approach enables the local client to use richer information anchor for learning, thereby alleviating contrastive learning performance degradation caused by distributed storage.

### C. SERVER AGGREGATION MODEL

When server receives the embeddings that clients have uploaded, it needs to update them jointly. We design an update mechanism with self-adaptive capabilities, as shown in 3. It aggregates similar embeddings, and iteratively adjusts the aggregation radius by computing the tightness score, until the optimal radius is found.

To evaluate the tightness through clustering effect, we introduce Davies-Bouldin Index (DB) and Calinski-Harabasz Index (CH) as the evaluation index of clustering.

*Definition 2:* Davies-Bouldin Index [51]

$$S_{e,D} = \frac{s(D)}{l(e, \mathrm{mean}(D))}$$

where $S_{e,D}$ is DB of an embedding $e$ and $D$, $D$ represents an embedding set and all embeddings in it are considered as a cluster, $s(D)$ is cluster $D$'s diameter, $mean(D)$ represents the cluster centre, $l(e, mean(D))$ represents the distance between $e$ and $mean(D)$.

DB measures intra-class closeness, a smaller value of $S_{e,D}$ means the more reasonable for $e$ to join the cluster.

*Definition 3:* Calinski-Harabasz Index [52]

$$CH(K) = \frac{B(K)(N-K)}{W(K)(K-1)}$$

where, $K$ is the number of clusters, $N$ is the total number of samples in clusters, $B(K)$ is inter-cluster divergence, $W(K)$ is intra-cluster divergence. CH measures the overall clustering effect, a higher CH indicates better global clustering.

The detailed process of server aggregation algorithm is as follows:

① First, for the embeddings received in the $k-th$ round, calculate their distances to others and arranged in descending order to obtain the Similarity Sequence:

$$SimSeq_{i,j} = \left\{e_1, e_2, \ldots, e_r, \ldots, e_{\left(\sum_{i=1}^{N}|V_i|\right)-1}\right\}$$

$e_r$ represents the embedding of a node with rank $r$ in $SimSeq_{i,j}$.

② The set $D_{i,j}$ of embeddings for updating $h^k_{i,j}$ is obtained recursively by sequentially querying embeddings from $SimSeq_{i,j}$.

*Definition 4:* Decision recursive formula. For selecting the optimal solution for updating a cluster from adjacent embeddings:

$$C(r) = \arg\min_{D} \left\{S_{e_r,C(r-1)}, S_{e_r,C(r-2)}\right\} \cup \left\{e_\gamma\right\}$$

where $C(0) = \left\{h^k_{i,j}\right\}, C(0) = \left\{h^k_{i,j}, e_1\right\}$.

Given a sensitive value $\varepsilon$, end the recursion when $\min_D\{S_{e_r,C(\gamma-1)}, S_{e_r,C(\gamma-2)}\} \geq \varepsilon$ to obtain $D_{i,j}$.

③ Perform mean pooling on the embedding in $D_{i,j}$ to update the initial embedding.

④ If $\varepsilon$ is small, sufficient information will not be obtained, and when $\varepsilon$ is large, the updates contain noisy information. To find the optimal sensitivity value, we first set a small $\varepsilon$, and cluster the updated embeddings at the end of all updates. In this paper, meanshift clustering [53] is used as an example. And then compute CH based on the clustering results.

Then, we gradually scaled up $\varepsilon$ and repeated the above process until the current clustering CH was smaller than the previous one. The previous $\varepsilon$ was considered as the optimal sensitivity value for this round.

The procedure of server aggregation is given in Algorithm 1. The time complexity of Algorithm 1 is $O(n^3 + n^2 \times m)$, where $n$ is the number of input embeddings, $m$ is the embedding's dimension.

**Algorithm 1:** Server Aggregation Algorithm.

```
 INPUT: training round $k$; number of clients, $N$; $k-th$
 round initial embedding set of server, $h^k$,
 $h^k = \{h^k_1, h^k_2, \ldots, h^k_i, \ldots, h^k_N\}$; $i-th$ client, $C_i$; $C_i$'s
 initial embedding, $h^k_i$,
 $h^k_i = \{h^k_{i,0}, h^k_{i,1}, \ldots, h^k_{i,j}, \ldots, h^k_{i,|V_i|}\}$; initial sensitivity
 value $\varepsilon$; amplification factor $\mu$; distance function $dis(\cdot)$;
 Initial round of iterations $r = 2$; Loop initial round
 $l = 1$; $CH_0 = CH_1 = 0$.
 OUTPUT: $k-th$ round updated embedding, $h^{k'}$;
 $h^{k'} = \{h^{k'}_1, h^{k'}_2, \ldots, h^{k'}_i, \ldots, h^{k'}_N\}$
# Get similarity sequence:
for $h^k_{i,j}$ in $h^k$ do
    for $h^k_{n,m}$ in $h^k$ do
        Calculate $dis\left(h^k_{i,j}, h^k_{n,m}\right)$, $h^k_{i,j} \neq h^k_{n,m}$
    end for
        Sorted in descending order according to
     calculations.
        Get $SimSeq \longleftarrow$
     $\{SimSeq_{1,0}, SimSeq_{1,1}, \ldots, SimSeq_{N,|V_N|}\}$,
        $SimSeq_{i,j} \longleftarrow$
     $\left\{e_1, e_2, \ldots, e_r, \ldots, e_{\left(\sum_{i=1}^{N}|V_i|\right)-1}\right\}$
end for

# Obtaining decision set for updating by recursion and
 mean pooling:
while $CH_l < CH_{l-1}$ do
    for $h^k_{i,j}$ in $h^k$ do
        while $\min_D\{S_{e_r,C_{i,j}(r-1)}, S_{e_r,C_{i,j}(r-2)}\} < \varepsilon$ do
            $C_{i,j}(r) \longleftarrow$
        $\arg\min_D\{S_{e_r,C_{i,j}(r-1)}, S_{e_r,C_{i,j}(r-2)}\} \cup \{e_r\}$
            $r \longleftarrow r + 1$
        end while
        $h^{l,k'}_{i,j} \longleftarrow mean(C_{i,j}(r-2))$
    end for
    # Optimal sensitivity value selection:
    meanshift on $h^{k'}$
    Calculate $CH_l$
    $l \longleftarrow l + 1$
    $\varepsilon \longleftarrow (1 + \mu)\varepsilon$
end while
return $h^{k'} \longleftarrow h^{l-2,k'}$
```

## V. ANALYSIS

In this section, we will conduct a communication security analysis as well as a framework efficiency analysis.

### A. SECURITY ANALYSIS

To avoid privacy leakage during communication, we use differential privacy in the communication process. In order to secure privacy while adding less noise and gaining more flexibility, we adopt Gaussian mechanism for differential privacy. Gaussian mechanism is defined as follows:

*Theorem 1:* Gaussian mechanism. [22] For a function $f : D \rightarrow R$ over a dataset $D$, the mechanism $M$ in (2) provides the -differential privacy.

$$M(D) = f(D) + Y \tag{2}$$

where Y is the Gaussian noise satisfying the (3) condition:

$$\forall \delta \in (0, 1), \sigma > \frac{\sqrt{2\ln(1.25/\delta)}\Delta f}{\varepsilon}, Y \sim N\left(0, \sigma^2\right) \tag{3}$$

For all transmitted data, the $(\epsilon, \delta)$-differential privacy defined in Definition 1 is satisfied by adding a Gaussian noise to it that meets the above conditions.

### B. EFFICIENCY ANALYSIS

In this section, we compare our method with vanilla method (GraphCL with FedAvg) in terms of communication efficiency and local storage space consumption to provide an analysis of the applicability scenarios of this framework.

#### 1) COMMUNICATION EFFICIENCY

To ensure a fair comparison, we set the neural network layers of GraphCL to be consistent with the framework of this paper. It is assumed that the local GraphCL of device $D_i$ has a total of m layers of network with parameter dimensions $F_i \times dim_1$, $dim_1 \times dim_2$,..., $dim_{m-1} \times dim_m$. The gradient has the same size as the parameters.

When the gradients of GraphCL are federated for learning with the FedAvg method, the size of the data that needs to be transferred for each round by device $D_i$ is:

$$F_i \times dim_1 + dim_1 \times dim_2 + \cdots + dim_{m-1} \times dim_m \tag{4}$$

Whereas SC-FGCL method passes node embedding, the size of data to be transmitted is:

$$N_i \times dim_m \tag{5}$$

According to (4) and (5), it can be found that our method is insensitive to feature dimensionality, the size of network parameters, and the number of nodes, while GraphCL is the opposite.

In the case of distributed storage, the number of nodes stored by each client tends to be significantly less than centralized case, so SC-FGCL can better support multiple feature data, support multi-layer networks, and the transmission communication is not limited by the network structure.

#### 2) LOCAL STORAGE SPACE CONSUMPTION

For device $D_i$ holding $G_i : \{N_i, E_i\}$, $E_i$ stored in the form of an adjacency matrix, the size of the storage space occupied by the origin graph is:

$$N_i \times F_i + N_i \times N_i \tag{6}$$

TABLE 2. Dataset

| Dataset | Cora | Citeseer | Pubmed |
|---|---|---|---|
| Nodes | 2708 | 19717 | 3327 |
| Edges | 5429 | 44338 | 4732 |
| Features | 1433 | 500 | 3703 |
| Classes | 7 | 6 | 3 |

GraphCL generally obtains two augmented graphs from the origin graph, and we discuss the storage consumption under three common augmentation approaches:

Node mask:

The storage consumption at node mask is:

$$N_{cost} = 2\alpha(N_i \times F_i + \alpha N_i \times N_i) + N_i \times F_i + N_i \times N_i \tag{7}$$

where $\alpha$ is the node retention probability and $\alpha \in (0, 1)$.

Feature mask:

The storage consumption at feature mask is:

$$F_{cost} = 2(\beta N_i \times F_i + N_i \times N_i) + N_i \times F_i + N_i \times N_i \tag{8}$$

where $\beta$ is the node retention probability and $\beta \in (0, 1)$.

Subgraph:

The storage consumption at subgraph is:

$$S_{cost} = 2\alpha(\beta N_i \times F_i + \alpha N_i \times N_i) + N_i \times F_i + N_i \times N_i \tag{9}$$

The only data that SC-FGCL needs to propagate through the network is the local origin graph, and the combined (6)–(9), SC-FGCL always consumes less local storage than GraphCL.

## VI. EXPERIMENTS

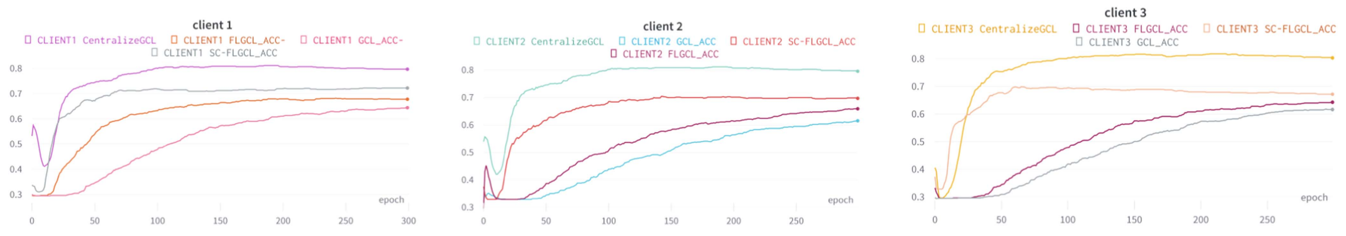
### A. DATASETS

Cora, Pubmed, and Citeseer [54] are three benchmark datasets commonly used in graph neural networks. We divide the node, edge, and feature data so that they are stored in different clients to simulate data isolation. To validate the effectiveness of GCL as a pre-training, we select node classification as the downstream task and use the downstream task accuracy as the evaluation criterion. Details of these three datasets are shown in Table 2.

### B. COMPARISON METHODS

As there is no systematic approach focused on mitigating the performance degradation of local GCL through a federal approach before our work. So we compared the downstream task accuracy for the following cases with the same training settings:

1) Centralized training: centralized training to obtain pretrain models
2) Local training: local training via GraphCL only
3) Baseline: network gradient averaging via FedAvg in each round while training locally via GraphCL
4) Our methods



(a) client 1 (b) client 2 (c) client 3

**FIGURE 4.** Accuracy of node classification in cora.

We compare the training results between the centralized training and the local training scenario with GraphCL to verify that data isolation reduces the effectiveness of GCL.We use GraphCL+FedAvg as a baseline to compare its training effect on the local client.

## C. PARAMETER SETTING

In our experiments, the total number of clients is 3; the $(\epsilon, \delta)$-differential privacy parameters $\epsilon \in 1, 2, 4, 8, 16, 32, 64, 128$, $\delta = 10^4$ are set; The learning rate for local GCL is 0.01; A multi-layer perceptron classifier is used for node classification, including a fully connected layer and a activation layer, with a Relu activation layer function and a learning rate of 0.01.

## D. ANALYSIS OF RESULTS

Table 3 shows the experimental results of the node classification task using different datasets. Furthermore, for visual analysis, we chose the results from the cora set and plotted their accuracy curves for all clients, as shown in Fig. 4.

As shown in Fig. 4, the prediction accuracy of centralised training is consistently significantly higher than the distributed training and converged faster. According to Table 3, the prediction accuracy of the centralised case is 10%–20% higher than the distributed one. This indicates that distributed storage of data can significantly reduce the effectiveness of GCL.

For baseline, gradients are averaged each round during the training process. According to the results shown in Table 3, FedAvg has a 1%–5% increase for the distributed GraphCL. And as shown by Fig. 4, convergence is slightly faster when FedAvg is used.

The SC-FGCL method has superior performance both in terms of accuracy and convergence speed. As shown in Table 3(a), the SC-FGCL has a 4%–7% improvement over baseline in the Cora set, and for the citeseer and pubmed sets, the SC-FGCL method results are close to centralized training results. The curves in Fig. 4 show that the SC-FGCL converges significantly faster than the other methods and is similar to the convergence speed of the centralised method. This demonstrates the significant role of the SC-FGCL method in mitigating the reduced GCL effect due to distributed storage, as well as its superior performance as a federal GCL framework.

**TABLE 3.** Accuracy of Node Classification Different Method

(a) Cora

| Dataset | Cora | | |
|---|---|---|---|
| **Device** | Client 1 | Client 2 | Client 3 |
| **Centralized** | 80.34%±0.02% | 79.63%±0.10% | 80.36%±0.15% |
| **GraphCL** | 66.14%±0.34% | 62.28%±1.41% | 61.09%±1.00% |
| **GraphCL+FedAvg** | 67.24%±0.72% | 65.30%±1.00% | 62.64%±1.12% |
| **SC-FGCL** | 71.80%±1.54% | 67.32%±2.21% | 69.01%±1.30% |

(b) Citeseer

| Dataset | Citeseer | | |
|---|---|---|---|
| **Device** | Client 1 | Client 2 | Client 3 |
| **Centralized** | 70.79%±0.02% | 71.93%±0.13% | 72.79%±0.12% |
| **GraphCL** | 61.61%±0.11% | 67.18%±0.11% | 66.73%±0.13% |
| **GraphCL+FedAvg** | 61.42%±0.32% | 67.71%±0.21% | 66.85%±0.11% |
| **SC-FGCL** | 66.37%±1.22% | 67.36%±1.32% | 69.53%±1.87% |

(c) Pubmed

| Dataset | Pubmed | | |
|---|---|---|---|
| **Device** | Client 1 | Client 2 | Client 3 |
| **Centralized** | 86.58%±0.06% | 85.47%±0.03% | 87.60%±0.02% |
| **GraphCL** | 77.17%±0.10% | 74.06%±0.11% | 75.92%±0.08% |
| **GraphCL+FedAvg** | 80.00%±0.42% | 79.14%±0.24% | 76.04%±0.17% |
| **SC-FGCL** | 82.39%±0.34% | 82.40%±0.73% | 84.51%±0.48% |

To verify the effectiveness of the server aggregation scheme proposed in this paper, we conducted ablation experiments on three datasets and selected the results of client1 for observational analysis. We shows the results in Table 4. The top10 embeddings in the similarity matrix are used to update the original embeddings when no regression decision is made and the optimal clustering sensitivity value selection is performed via CH.

According to Table 3, the CH-index binding-only approach outperforms the no-strategy approach, while the regression decision-only approach outperforms the CH-index binding-only approach. And is optimal when both approaches are taken. This proves that regression decision method has a more

**TABLE 4.** Ablation Experiments

| Recursion Decision | CH-index Binding | Accuracy | | |
|---|---|---|---|---|
| Dataset | | cora | citeseer | pubmed |
| √ | √ | 71.80%±1.5% | 66.37%±1.2% | 82.39%±0.3% |
| √ | × | 69.73%±1.2% | 66.12%±0.7% | 82.14%±2.0% |
| × | √ | 69.19%±3.3% | 65.78%±2.0% | 80.22%±10.1% |
| × | × | 68.82%±4.2% | 65.32%±5.1% | 79.81%±13.2% |

significant utility and that CH-index binding acts as an adjustment scheme to assist the regression decision method in achieving the optimal value.

## VII. CONCLUSION

In this paper, we propose a Self-adaptive Cluster-based Federal Graph Contrastive Learning (SC-FGCL) framework which unites all clients to update node embeddings as local GCL anchors to mitigate the weakening of GCL effects by distributed stored graph data, resulting in better GCL effects. At the same time, to enable the server to find the optimal update solution, we designed clustering methods with self-adaptive capabilities. It allows each embedding to be updated in a way that maximizes the enrichment of its own information while preventing noisy embeddings from interfering. Experimental results on multiple graph datasets show that our method significantly outperforms the comparative baseline and that our self-training approach yields better performance.

## REFERENCES

[1] C. Xia, S.-H. Feng, Y. Xia, X. Pan, and H.-B. Shen, "Fast protein structure comparison through effective representation learning with contrastive graph neural networks," *PLoS Comput. Biol.*, vol. 18, no. 3, 2022, Art. no. e1009986.

[2] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive representation learning: A framework and review," *IEEE Access*, vol. 8, pp. 193907–193934, 2020.

[3] Y. You, T. Chen, Y. Shen, and Z. Wang, "Graph contrastive learning automated," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12121–12132.

[4] D. Xu, W. Cheng, D. Luo, H. Chen, and X. Zhang, "InfoGCL: Information-aware graph contrastive learning," in *Adv. Neural Inf. Process. Syst.*, vol. 34, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds. 2021, pp. 30414–30425. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/ff1e68e74c6b16a1a7b5d958b95e120c-Paper.pdf

[5] P. Wang, K. Han, X.-S. Wei, L. Zhang, and L. Wang, "Contrastive learning based hybrid networks for long-tailed image classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 943–952.

[6] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 5812–5823, 2020.

[7] P. Veličković et al., "Deep graph infomax," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–17.

[8] S. Li, X. Wang, A. Zhang, Y. Wu, X. He, and T.-S. Chua, "Let invariant rationale discovery inspire graph contrastive learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 13052–130 65.

[9] J. Xia, L. Wu, G. Wang, J. Chen, and S. Z. Li, "ProGCL: Rethinking hard negative mining in graph contrastive learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 24332–24346.

[10] L. Zhang, M. Li, K. Yan, R. Wang, and B. Hui, "Hierarchical knowledge-based graph embedding model for image-text matching in IoTs," *IEEE Internet of Things J.*, vol. 9, no. 12, pp. 9399–9409, Jun. 2022.

[11] L. Zhang, S. Feng, G. Duan, Y. Li, and G. Liu, "Detection of microaneurysms in fundus images based on an attention mechanism," *Genes*, vol. 10, no. 10, 2019, Art. no. 817.

[12] Q. Chen, Z. Cai, L. Cheng, F. Wang, and H. Gao, "Joint near-optimal age-based data transmission and energy replenishment scheduling at wireless-powered network edge," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2022, pp. 770–779.

[13] Q. Chen, Z. Cai, L. Cheng, and H. Gao, "Structure-free general data aggregation scheduling for multihop battery-free wireless networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 9, pp. 3342–3359, Sep. 2022.

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

[15] X. Ni, X. Xu, L. Lyu, C. Meng, and W. Wang, "A vertical federated learning framework for graph convolutional network," 2021, *arXiv:2106.11593*.

[16] Z. He and J. Zhou, "Inference attacks on genomic data based on probabilistic graphical models," *Big Data Mining Anal.*, vol. 3, no. 3, pp. 225–233, 2020.

[17] Y. Liang, Z. Cai, J. Yu, Q. Han, and Y. Li, "Deep learning based inference of private information using embedded sensors in smart devices," *IEEE Netw.*, vol. 32, no. 4, pp. 8–14, Jul./Aug. 2018.

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

[19] J. Zhang and Q. Xu, "Attention-aware heterogeneous graph neural network," *Big Data Mining Anal.*, vol. 4, no. 4, pp. 233–241, 2021.

[20] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.

[21] W. Zhang, Z. Li, and X. Chen, "Quality-aware user recruitment based on federated learning in mobile crowd sensing," *Tsinghua Sci. Technol.*, vol. 26, no. 6, pp. 869–877, 2021.

[22] T. Zhu, G. Li, W. Zhou, and S. Y. Philip, "Differentially private data publishing and analysis: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 8, pp. 1619–1638, Aug. 2017.

[23] X. Zheng and Z. Cai, "Privacy-preserved data sharing towards multiple parties in industrial IoTs," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 968–979, May 2020.

[24] X. Zheng, L. Zhang, K. Li, and X. Zeng, "Efficient publication of distributed and overlapping graph data under differential privacy," *Tsinghua Sci. Technol.*, vol. 27, no. 2, pp. 235–243, 2022.

[25] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–17.

[26] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–16.

[27] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.

[28] K. Sohn, "Improved deep metric learning with multi-class N-pair loss objective," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 1857–1865.

[29] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv:1807.03748*.

[30] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3733–3742.

[31] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proc. Web Conf. 2021*, 2021, pp. 2069–2080.

[32] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 4116–4126.

[33] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 12, pp. 11015–11023.

[34] Y. You, T. Chen, Z. Wang, and Y. Shen, "Bringing your own view: Graph contrastive learning without prefabricated data augmentations," in *Proc. 15th ACM Int. Conf. Web Search Data Mining*, 2022, pp. 1300–1309.

[35] J. Qiu et al., “GCC: Graph contrastive coding for graph neural network pre-training,” in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1150–1160.
[36] M. Xu, H. Wang, B. Ni, H. Guo, and J. Tang, “Self-supervised graph-level representation learning with local and global structure,” in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 11548–11558.
[37] H. Zhao, X. Yang, Z. Wang, E. Yang, and C. Deng, “Graph debiased contrastive learning with joint representation clustering,” in *Proc. Int. Joint Conf. Artif. Intell.*, 2021, pp. 3434–3440.
[38] G. Chu, X. Wang, C. Shi, and X. Jiang, “CuCo: Graph representation with curriculum contrastive learning,” in *Proc. Int. Joint Conf. Artif. Intell.*, 2021, pp. 2300–2306.
[39] J.-B. Grill et al., “Bootstrap your own latent-a new approach to self-supervised learning,” *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 21271–21284, 2020.
[40] N. Lee, J. Lee, and C. Park, “Augmentation-free self-supervised learning on graphs,” in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 7, pp. 7372–7380.
[41] S. Thakoor, C. Tallec, M. G. Azar, R. Munos, P. Veličković, and M. Valko, “Bootstrapped representation learning on graphs,” in *Proc. ICLR Workshop Geometrical Topological Representation Learn.*, 2021.
[42] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” *Proc. Mach. Learn. Syst.*, vol. 2, pp. 429–450, 2020.
[43] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, “Federated learning with matched averaging,” in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–16.
[44] S. Hardy et al., “Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption,” 2017, *arXiv:1711.10677*.
[45] K. Wei et al., “Federated learning with differential privacy: Algorithms and performance analysis,” *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3454–3469, 2020.
[46] A. N. Bhagoji, S. Chakraborty, P. Mittal, and S. Calo, “Analyzing federated learning through an adversarial lens,” in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 634–643.
[47] M. Chen, W. Zhang, Z. Yuan, Y. Jia, and H. Chen, “Federated knowledge graph completion via embedding-contrastive learning,” *Knowl.-Based Syst.*, vol. 252, 2022, Art. no. 109459.
[48] H. Yang, X. Zhao, M. Li, H. Chen, and G. Xu, “Federated graph contrastive learning,” 2022, *arXiv:2207.11836*.
[49] Z. Cai, Z. He, X. Guan, and Y. Li, “Collective data-sanitization for preventing sensitive information inference attacks in social networks,” *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 4, pp. 577–590, Jul./Aug. 2018.
[50] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, “Our data, ourselves: Privacy via distributed noise generation,” in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Techn.*, Berlin, Germany: Springer, 2006, pp. 486–503.
[51] D. L. Davies and D. W. Bouldin, “A cluster separation measure,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 2, pp. 224–227, Apr. 1979.
[52] T. Caliński and J. Harabasz, “A dendrite method for cluster analysis,” *Commun. Statist.-Theory Methods*, vol. 3, no. 1, pp. 1–27, 1974.
[53] K. Fukunaga and L. Hostetler, “The estimation of the gradient of a density function, with applications in pattern recognition,” *IEEE Trans. Inf. Theory*, vol. 21, no. 1, pp. 32–40, Jan. 1975.
[54] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, “Collective classification in network data,” *AI Mag.*, vol. 29, no. 3, pp. 93–93, 2008.