Contrastive and Non-Contrastive Strategies for Federated Self-Supervised Representation Learning and Deep Clustering

Runxuan Miao and Erdem Koyuncu, *Senior Member, IEEE*

***Abstract*—We investigate federated self-supervised representation learning (FedSSRL) and federated clustering (FedCl), aiming to derive low-dimensional representations of datasets distributed across multiple clients, potentially in a heterogeneous manner. Our proposed solutions for both FedSSRL and FedCl involves a comparative analysis from a broad learning context. In particular, we show that a two-stage model, beginning with representation learning and followed by clustering, is an effective learning strategy for both tasks. Notably, integrating a contrastive loss as regularizer significantly boosts performance, even if the task is representation learning. Moreover, for FedCl, a contrastive loss is most effective in both stages, whereas FedSSRL benefits more from a non-contrastive loss. These findings are corroborated by extensive experiments on various image datasets.**

***Index Terms*—Federated learning, representation learning, clustering.**

I. Introduction

A. Federated Learning and Representation Learning

Federated learning involves training a global machine learning model across multiple decentralized devices or servers. In each round, local models on individual devices are trained on their own datasets, and then the models' parameters are aggregated and averaged to update the global model. In the server aggregation stage of federated learning, a central server collects all local updates from clients to obtain a global model, generally through weighted averaging of the local models. It has various applications in areas such as finance, healthcare, and autonomous vehicles, particularly in scenarios where considerations of privacy, data decentralization, and computational capacity are crucial. There are numerous algorithms [1]–[4] designed for federated learning, such as federated averaging [3]. However, the existing literature primarily concentrates on supervised learning, while we shall focus on unsupervised learning.

Self-supervised representation learning [5]–[8] aims to create meaningful representations of data without requiring labels for training, which can then be applied to downstream tasks such as linear classification, object detection, semi-supervised learning and clustering. The representation space typically has much smaller dimension than the data space. Two main approaches include contrastive [5] and non-contrastive [6] learning. The contrastive loss leverages large negative samples to encourage representations of similar data augmentations to be close. In contrastive learning, two identical networks process two different augmented views of the same input. In contrast, non-contrastive learning utilizes a mean squared error (MSE) loss in a dual-network architecture with online and target networks, eliminating the need for explicit negatives. However, whether using contrastive or non-contrastice learning, most state-of-the-art methods assume a centralized data setting, making them unsuitable for distributed data scenarios.

B. Federated Self-Supervised Representation Learning (FedSSRL) and Federated Clustering (FedCl)

Recently, several works have studied the FedSSRL problem [9]–[12], where each local model is trained on unlabeled data using contrastive or non-contrastive loss functions, as described above [13]. The objective is to obtain a global model that produces meaningful representations for downstream tasks, such as linear evaluation and semi-supervised learning. In our study, we focus not only on FedSSRL, but also on the particular downstream task of clustering. In this Federated Clustering (FedCl) framework, the global model is specifically trained to directly produce the clustering assignments.

Clustering is a machine learning and data analysis technique wherein data points are grouped together based on certain similarities or patterns in an unsupervised manner. The problem is intimately related to representation learning. In fact, representation learning aims to obtain a lower-dimensional representation of a higher-dimensional input via a continuous mapping. Clustering

The authors are with the Department of Electrical and Computer Engineering, University of Illinois at Chicago. E-mails: {rmiao6, ekoyuncu}@uic.edu. This work was presented in part at the IEEE ICASSP Worskhop on Timely and Private Machine Learning over Networks, June 2023, ICML Workshop on Federated Learning and Analytics in Practice: Algorithms, Systems, Applications, and Opportunities, July 2023, and at IEEE International Workshop on Machine Learning for Signal Processing, Sept. 2023. This work was supported in part by the Army Research Lab (ARL) under Grants W911NF-2120272 and W911NF-2420172, by National Science Foundation (NSF) under Grant CNS-2148182, and by the Army Research Office (ARO) under Grant W911NF-2410049.

can be thought as an extreme form of representation learning where the mapping is discrete and the input should thus be represented by only finitely many points. A fundamental problem that we will address in the present work is how optimal deep learning algorithms will change as we transition from federated unsupervised representation learning (continuous mappings) to federated clustering (finite discrete mappings).

Clustering has been applied in various domains, including anomaly detection, data compression, social network analysis, and recommendation systems [14]. K-Means [15], a conventional clustering algorithm, provides efficient performance in many data clustering tasks. However, it struggles to effectively differentiate between clusters in high-dimensional, real-world datasets.

With the advancement of deep learning techniques, deep clustering [16]–[19] has emerged as a trend for jointly optimizing clustering assignments and learning data representations, allowing for more effective clustering in high-dimensional spaces. For example, Deep Embedded Clustering (DEC) [18] optimizes the cluster centers and determines the clustering assignment for each sample by evaluating the similarity between the data representation and each cluster center. However, the effectiveness of DEC is constrained to relatively simple datasets and does not extend well to large-scale and complex ones. Moreover, an increasing number of deep clustering frameworks are constructed upon SSRL. For instance, contrastive clustering (CC) [16] generates soft labels from the feature matrix by incorporating a cluster-level contrastive loss. However, the majority of state-of-the-art deep clustering approaches are currently centralized [20]. Surprisingly, despite being a research problem of fundamental importance, the FedCl task has not been thoroughly investigated in the literature, with only a few studies focusing on specific datasets, such as medical datasets [21], or on data with simple features [22]–[25].

The goal of federated clustering is to learn a global model that generates clustering assignments. Applications of clustering can serve as practical scenarios for our proposed model. For example, clustering can be used in market research to segment customers based on their purchasing behaviors and preferences, assisting businesses in adjusting marketing strategies and products to different customers. However, the data is distributed across multiple sources, such as mobile applications, so that it is important to keep the data private and train the model locally. Thus, federated clustering helps in providing personalized experiences and recommendations by local training while maintaining the user privacy.

C. Contributions of the Paper

The primary questions we aim to address in this paper are as follows: **Q1:** Which learning approach, contrastive or non-contrastive, is most appropriate for FedSSRL and FedCl? **Q2:** Building on Q1's exploration, how do we precisely configure the chosen approach to maximize its effectiveness in each respective federated paradigm?

At this juncture, elaborating on Q1 becomes insightful: The downstream characteristic of deep clustering tasks indicates that contemporary methods encompass two distinct phases – an initial representation learning stage followed by clustering. For a comprehensive view, we also examine two-stage methodologies for FedSSRL, where the clustering phase serves as a regularizer.

Addressing **Q1**, our main finding in this paper is that two-stage methodologies are most effective for both FedSSRL and FedCl. Specifically, employing a non-contrastive learning strategy at both stages yields the best results for FedSSRL, whereas a contrastive learning approach at each stage is found to be ideal for FedCl. An intuitive explanation, which is further substantiated by numerous numerical simulations in our study, is as follows: Contrastive learning excels in generating a uniform distribution of representations, owing to its extensive use of negative samples. However, it often encounters the challenge of class collision, where samples from the same class within a batch are mistakenly treated as negative samples. Conversely, non-contrastive learning adeptly circumvents the class collision problem. Nevertheless, it tends to produce non-uniform representations, which often leads to the disproportionate assignment of the majority of samples to a limited number of clusters. This issue undermines the clustering process, as it results in a highly non-uniform distribution of samples across different clusters.

After establishing the broadly optimal learning approach through **Q1**, our attention in **Q2** turns to the design of the learning scheme, taking into account whether a non-contrastive or contrastive learning method is utilized. First, for the FedSSRL problem, where a non-contrastive approach is optimal, we introduce Federated Representation Learning through Clustering (FedRLC). The key novelty of FedRLC is to utilize a clustering loss as a regularizer to boost the representation learning performance. In addition, we consider a data selection strategy for training; not all samples are considered in the clustering loss function. Specifically, we use an input data for training when its predicted labels from the online network and the target network are the same, as they are expected to output similar embeddings. This increases the likelihood of selecting data with more accurate soft labels for training. FedRLC also incorporates a new dynamic controller to update the cluster centers in server to client communications. Next, we focus on the FedCl

problem, and introduce Federated Clustering through Past Negatives Pool (FedPNP), based on contrastive learning. The key novelty of FedPNP is to use the negatives pool from the past versions of local models to avoid the client drift and class collision problems in federated learning. We show through numerical simulations that both FedRLC and FedPNP schemes achieve state-of-the-art results in various image datasets.

The main goal of the proposed schemes is to maximize the eventual classification or clustering accuracy of schemes. Of course, it is also desirable to minimize the convergence time of the learning process. In this context, we note that our proposed schemes can be combined with other complementary techniques that improve convergence such as dataset pruning [26], [27] or model-distributed learning [28], [29].

Parts of this work was presented in [30]–[32]. In comparison to these works, this paper focuses on the joint study of FedRL and FedCl, with a particular focus on **Q1**. In other words, we first seek to determine the specific learning approach (contrastive or non-contrastive) that is suitable for the particular task. The rest of this paper is organized as follows: Section II provides an in-depth review of the related work. The preliminaries of our work are presented in Section III. We optimize our schemes over the contrastive/non-contrastive learning space in Section IV. Section V introduces our proposed scheme for represenation learning, FedRLC. Section VI details our novel federated clustering scheme, FedPNP. Evaluation and ablation of the proposed models are provided in Sections VII and VIII, respectively. Finally, we conclude the paper in Section IX.

II. Related Work

A. Federated Learning

Federated learning proves to be valuable in situations where the data sharing is constrained, and the computational capacity is restricted. Meanwhile, it introduces challenges such as communication overhead during model transmission and client drift caused by non-iid data. Many federated learning algorithms [1]–[4], [33]–[35] have been developed to address these issues. For example, the classical federated averaging [3] aggregates model updates from decentralized client devices, and FedProx [2] modifies federated averaging by including a regularization term to enhance convergence and improve global model performance. However, most algorithms are designed for supervised learning, where labeled data is necessary for training, which may not always be practical. In this study, we investigate how to achieve high-quality low-dimensional embeddings of distributed unlabeled data through federated learning.

B. Centralized and Federated SSRL and Clustering

Recently, SSRL has garnered significant attention for its ability to learn representations without relying on label information [5]–[8], [36], [37]. In the realm of SSRL, contrastive and non-contrastive loss functions have become the favored choices for learning instance representations. Concurrently, there is a growing trend in the development of deep clustering techniques, as evidenced by various studies like those by [38]–[41]. These techniques primarily utilize either contrastive or non-contrastive loss functions, paving the way for clustering methods that are grounded in SSRL such as [16], [42].

Contrastive learning has been well-studied in SSRL [5], [6], [43] and deep clustering [19], [44], [45], but it suffers from class collision, where positive samples from the same class with the given input are still viewed as negatives in contrastive loss. Two main directions of study are sampling positives [19], [44] and selecting negatives [46], [47]. For example, GCC [19] and WCL [45] expand positive samples by building graphs. MoCHi [47] chooses the hardest negatives by sorting the instance similarity with the given query based on the dot product similarity. However, sorting high dimensional instance embeddings and constructing graphs [19], [48] consume time and computational resources. Moreover, these strategies assume that the data is central.

The main challenges in federated learning that we seek to overcome are as follows: First, we wish to learn from distributed data as opposed to centralized data subject to limited communication rates between the clients and the server. Second, we assume that the data across multiple clients is not an independent and identically distributed (IID) random variable in general. This Non-IID assumption leads to significant divergence between local and global models, which is commonly referred to as the client drift problem. Also, the clients do not have enough representatives of different classes, significantly complicating the local as well as the global learning processes.

Different techniques have been proposed to remedy the client drift problem caused by Non-IID data. For example, SCAFFOLD [1] introduces a control variate for client updates. However, it is mainly used in supervised federated learning, and thus not directly applicable in our settings. FedGrEM [49], which was submitted after earlier versions of this work had been published, focuses on theoretical results on federated unsupervised learning on Non-IID data with experiments on simple datasets such as MNIST and Fashion MNIST. Here, we focus on practical aspects of both federated self-supervised learning and deep clustering on more complex datasets such as CIFAR-100. To address the significant performance drop in Non-IID data, FedRLC updates cluster centers based on the KL divergence between probabilities from

global and local networks, while FedPNP introduces a past negative pool to select data from previous iterations. Experimental results demonstrate that our proposed models effectively handle Non-IID scenarios in practice.

C. Federated SSRL and Clustering

In recent years, many works [9]–[11] have studied SSRL in the context of federated learning. For example, FedEMA [10] incorporates a non-contrastive loss in its local federated learning and adopts a weight divergence strategy for updating the encoder and predictor of the BYOL [6] network. It should be noted that although several papers address federated clustering, they predominantly concentrate on the clustering of clients [50]–[52]. In contrast to these studies, our work is exclusively focused on the clustering of data rather than clients. For instance, the recently introduced FeatARC [11] implements an additional loss function in local training phases and deploys a client clustering strategy to improve the performance of FedSSRL. Specifically, clients are grouped into clusters, and local models are updated based on the model of their respective cluster instead of a single global model. In our study, we explicitly consider clustering data, with the goal of learning a global model that automatically generates clustering assignments. While certain studies employ traditional K-means for data clustering in federated learning, this method is typically constrained to simpler datasets like MNIST and often struggles with more complex datasets. In this study, we explore the performance of SSRL-based deep clustering methods when applied to distributed datasets within the federated learning paradigm.

III. Preliminaries

The primary objective of FedSSRL is to obtain a global model that is able to generate meaningful data embeddings. These are subsequently employed in downstream tasks such as linear evaluation. On the other hand, FedCl aims to training a global model that assigns data points to appropriate clusters. Both methods essentially aim to learn low dimensional embeddings.

Suppose there are K clients, where Client k has its local unlabeled data $\mathcal{D}_k$. Our goal is to learn a model over the dataset $\mathcal{D} \triangleq \bigcup_{k=1}^{K} \mathcal{D}_k$ on a central server. We learn a global model by training and aggregating models trained at each client with local data. In this section, we consider the SSRL loss as serving for instance representation learning, while the clustering loss is specifically utilized for clustering assignments. First, we present the contrastive and the non-contrastive loss functions that are commonly utilized in the context of SSRL.

A. Self-supervised Representation Learning

Contrastive loss [5]: We first introduce the contrastive loss function, as considered in SimCLR [5]. Each input example is transformed into multiple augmented views through data augmentation techniques. The model then learns to bring positive representations closer together while pushing negative representations apart in the learned feature space. Specifically, we consider a dataset $\mathcal{D} = \{x_1, \ldots, x_{|\mathcal{D}|}\}$. Given an input $x_i \in \mathcal{D}$, the SimCLR scheme first creates two samples $x_i^a \triangleq t^a(x_i)$ and $x_i^b \triangleq t^b(x_i)$ through transformations t^a and t^b, respectively. We use the variable $\sigma \in \{a, b\}$ to represent the sample index so that the transformations are succinctly expressed as $x_i^\sigma \triangleq t^\sigma(x_i)$, $\sigma \in \{a, b\}$. The transformations are sampled uniformly at random from a family $\mathcal{T}$ of augmentations, which may include rotations, noise, etc. The samples then pass through the same encoder f, creating feature vectors $h_i^\sigma \triangleq f(x_i^\sigma)$, $\sigma \in \{a, b\}$. An instance-level multi-layer perceptron (MLP) g_I projects h_i^a and h_i^b to obtain instance-level representations $z_i^\sigma \triangleq g_I(h_i^\sigma) \in \mathbb{R}^{d_1}$, $\sigma \in \{a, b\}$, where d_1 represents the dimension of the obtained representations.

The similarity of any two representations is compared via the cosine similarity measure $s(u, v) \triangleq u^\dagger v/(\|u\|\|v\|)$. To define the loss functions, we need the following definitions. Given matrices $\mathbf{u} = [u_1 \cdots u_n] \in \mathbb{R}^{d\times n}$ and $\mathbf{v} = [v_1 \cdots v_n] \in \mathbb{R}^{d\times n}$ constructed via the indicated column vectors, we define the ordinary contrastive loss function

$$(f, g_I) \mapsto L(\mathbf{u}, \mathbf{v}; \tau) \triangleq \frac{1}{n}\sum_{i=1}^{n} -\log \frac{\exp\left(\frac{1}{\tau}s(u_i, v_i)\right)}{\sum_{\substack{j=1\\ j\neq i}}^{n}\left[\exp\left(\frac{1}{\tau}s(u_i, u_j)\right)+\exp\left(\frac{1}{\tau}s(u_i, v_j)\right)\right]}. \tag{1}$$

Given a batch size n, the contrastive loss in SimCLR L_{C} is then defined via the instance-level representations $\mathbf{z}^\sigma \triangleq [z_1^\sigma \cdots z_n^\sigma] \in \mathbb{R}^{d_1\times n}$, $\sigma \in \{a, b\}$ as $L(\mathbf{z}^a, \mathbf{z}^b; \tau_I)$, where $\tau_I > 0$ is the instance-level temperature parameter

$$L_{\mathrm{C}} \triangleq L(\mathbf{z}^{\mathbf{a}}, \mathbf{z}^{\mathbf{b}}; \tau_I). \tag{2}$$

Non-contrastive loss [6]: The non-contrastive loss function is widely used in many frameworks, such as BYOL [6] and SimSiam [7]. The main differences between the contrastive and non-contrastive frameworks are that there are no negative samples while utilizing non-contrastive learning, and two augmented data instances are passed to two different networks: one referred to as the online network and the other as the target network. In detail, the online network consists of an online encoder f^O and an online predictor g^O, which are trained by gradient descent. We refer to the composition of the encoder and the projector in the original BYOL work as simply the "encoder" in this paper. The target

network only consists of a target encoder f^T. The weights of f^T are updated via the exponential moving average (EMA) of the online encoder f^O, as will be explained in the following. Given $\sigma \in \{a, b\}$ representing two augmentations, let $z_i^{\sigma,O} \triangleq g^O\left(f^O(t^\sigma(x_i))\right)$ and $z_i^{\sigma,T} \triangleq f^T(t^\sigma(x_i))$ denote the d-dimensional representations that one would obtain from the online and the target networks, respectively. Defining the scaled cosine similarity loss function as $\delta(x, y) \triangleq 2 - 2\frac{x^T y}{\|x\|\|y\|}$, BYOL uses the symmetrized loss

$$x_i \mapsto L_{\text{MSE}}(x_i) \triangleq \delta(z_i^{a,O}, z_i^{b,T}) + \delta(z_i^{b,O}, z_i^{a,T}). \quad (3)$$

The non-contrastive loss can then be defined as

$$(f^O, g^O) \mapsto L_{\text{NC}} \triangleq \sum_{x_i \in \mathcal{D}} L_{\text{MSE}}(x_i), \quad (4)$$

which signifies that only the online networks f^O, g^O are updated via gradient descent. The target network parameters are instead updated through the EMA

$$f^T \leftarrow \zeta f^T + (1 - \zeta) f^O, \quad (5)$$

where $\zeta \in [0, 1]$.

B. Deep clustering

We now consider the existing contrastive and non-contrastive-based clustering approaches; namely Deep Embedded Clustering (DEC) [18] and Contrastive Clustering (CC) [16]. Note that DEC is a non-contrastive algorithm, while CC is a contrastive clustering algorithm. The extension of these methods to the federated learning setting is elaborated in the next Section IV.

CC [16]: CC incorporates an instance-level contrastive loss and a cluster-level contrastive loss on the row and column space of the feature matrix, respectively. The primary idea of clustering in CC involves interpreting each row of the feature matrix as the soft labels of the instance, which are optimized concurrently by both the instance-level and cluster-level contrastive losses. The instance-level representation loss is the same as the SimCLR loss (2). Here, we provide their cluster-level contrastive loss. Similar to the instance-level MLP g_I in SimCLR, a cluster-level MLP g_C produces cluster-level representations $y_i^\sigma \triangleq g_C(h_i^\sigma) \in \mathbb{R}^{d_2}$, $\sigma \in \{a, b\}$, where h_i^σ is the data embedding from the same encoder f in SimCLR. In particular, in a deterministic assignment of inputs to clusters, all rows would be one-hot encoded vectors. On the other hand, given $\mathbf{c}^\sigma \triangleq [y_1^\sigma \cdots y_n^\sigma]^\dagger \in \mathbb{R}^{n \times d_2}$, $\sigma \in \{a, b\}$, the cluster-level contrastive loss is defined by $L_{\text{C}}(\mathbf{c}^a, \mathbf{c}^b; \tau_C)$, where $\tau_C > 0$ is the cluster-level temperature. In the CC scheme, the output dimensionality d_2 of the cluster-level representations is chosen to be equal to the number of clusters one wishes to find in the dataset. In many cases, the instance-level output dimensionality d_1 is chosen to be much larger than d_2. On the other hand, rows of $\mathbf{c}^a$ and $\mathbf{c}^b$ (i.e. y_i^σs) correspond to the soft labels of samples. In precise form, the cluster-level of CC loss function is given by

$$L_{C_1} \triangleq L(\mathbf{c}^a, \mathbf{c}^b; \tau_C) + H(\mathbf{c}^a) + H(\mathbf{c}^b), \quad (6)$$

where, for any matrix $\mathbf{u} = [u_1 \cdots u_d] \in \mathbb{R}^{n \times d}$, the entropy is defined as

$$H(u) \triangleq -\sum_{i=1}^{d} \frac{\|u_i\|_1}{\|\mathbf{u}\|_1} \log \frac{\|u_i\|_1}{\|\mathbf{u}\|_1}. \quad (7)$$

As discussed in [16], entropy regularization helps avoid the trivial solution where all samples are assigned to the same cluster.

DEC [18]: The original DEC [18] scheme consists of two stages. In the first stage, a stacked auto-encoder is pre-trained to produce data embeddings utilizing a reconstruction loss, specifically the least squares loss, which in our approach is substituted by the SSRL loss. In this paper, we focus on the second stage of the DEC work, namely the clustering assignment stage. Specifically, let $\mu_1, \ldots, \mu_M \in \mathbb{R}^d$ denote cluster centers, given the data x_i and its representation $z_i \in \mathbb{R}^d$, $q_{i,m}$ denotes the probability that the representation z_i belongs to cluster m with center μ_m. Following DEC [18], we model these cluster assignment probabilities with a student t-distribution with one degree of freedom

$$\Delta_m(z, \{\mu_n\}_{n=1}^M) \triangleq \frac{(1 + \|z - \mu_m\|^2)^{-\frac{1}{2}}}{\sum_n (1 + \|z - \mu_n\|^2)^{-\frac{1}{2}}}. \quad (8)$$

Specifically, we set

$$q_{i,m} = \Delta_m(z_i, \{\mu_n\}_{n=1}^M),\ m \in \{1, \ldots, M\},\ \forall i. \quad (9)$$

Effectively, each representation is assigned a probability distribution. According to (8), the closer the representation to a cluster center with index (say) m, the higher the belief/probability that the corresponding sample should belong to Cluster m. We now define a target distribution of the probabilities $q_{i,m}$ following [18], and we set

$$p_{i,m} = \frac{(q_{i,m})^2 / \sum_i q_{i,m}}{\sum_n \left[(q_{i,n})^2 / \sum_i q_{i,n}\right]}. \quad (10)$$

The target distribution is computed by squaring the probability and normalizing it by the frequency of each class. The motivation of squaring is to "harden" the soft assignments, while frequency normalization penalizes imbalanced clusters. The KL divergence loss is defined to compare the probability distributions. Letting

$$\text{KL}(p||q) \triangleq \sum_m p_m \log \frac{p_m}{q_m}, \quad (11)$$

the non-contrastive clustering loss is typically a KL divergence loss, which can be defined as

$$L_{\text{NC}_1} \triangleq \frac{1}{N} \sum_i \text{KL}(p_i||q_i). \quad (12)$$

IV. An Optimization over the Learning Space

A. The Two Stage Approach to Learning

In the previous section, we have presented two instance-level loss functions, (2) and (4), along with two cluster-level clustering loss functions, (6) and (12). For the purposes of representation learning, utilizing either a contrastive loss function as in (2) or a non-contrastive loss as in (4) is adequate. However, since clustering is typically considered a downstream task of representation learning, the initial step involves selecting a network for representation learning, followed by choosing a network for the downstream task that converts these representations into clusters. Given that there are two options available for both networks—contrastive and non-contrastive—we have four total combinations to consider. In fact, a novelty of the current work is that we also consider **a second clustering stage** for representation learning that will act as a regularizer, which will be thrown away once the training is complete. In the following, we describe the four possibilities that we optimize over:

Contrastive+Contrastive (C+C): As described in Section III, CC regularizes the representation contrastive loss (2) with a cluster-level contrastive loss (6). Thus, the overall CC loss is the sum of the two losses

$$L_{\mathrm{C+C}} = L_{\mathrm{C}} + L_{\mathrm{C}_1}. \tag{13}$$

This is, in fact, the loss function utilized in the CC framework [16].

Contrastive+Non-Contrastive (C+NC): Here, we simultaneously train the neural network and optimize the cluster centers by employing both the contrastive loss (2) and the non-contrastive KL divergence loss (12). Given the data x_i and the representation $z_i^\sigma, \sigma \in \{a, b\}$ from the SimCLR encoder f, let $q_{i,m}^\sigma = \Delta_m(z_i^\sigma, \{\mu_n\}_{n=1}^M)$, $m \in \{1, \ldots, M\}$, $\forall i$, denote the probability that the representation z_i^σ belongs to cluster m with center μ_m. Then, the target distribution

$$p_{i,m}^\sigma = \frac{(q_{i,m}^\sigma)^2 / \sum_i q_{i,m}^\sigma}{\sum_n \left[(q_{i,n}^\sigma)^2 / \sum_i q_{i,n}^\sigma\right]}. \tag{14}$$

Similar to (12), the KL divergence objective for two augmented views can be defined as

$$L_{\mathrm{NC}_2} \triangleq \frac{1}{n} \sum_i \Big[\mathrm{KL}(p_i^a || q_i^a) + \mathrm{KL}(p_i^b || q_i^b)\Big], \tag{15}$$

The instance-level loss remains consistent with the contrastive loss (2), so that the local loss of C+NC is

$$L_{\mathrm{C+NC}} = L_{\mathrm{C}} + L_{\mathrm{NC}_2}. \tag{16}$$

Non-Contrastive+Contrastive (NC+C): We include an additional cluster-level projector g_C following the online encoder f^O to get $y_i^{\sigma,O} \triangleq g_C(f^O(x_i^\sigma)) \in \mathbb{R}^{d_2}$, $\sigma \in \{a, b\}$ and $\mathbf{c}^{\sigma,O} \triangleq [y_1^{\sigma,O} \cdots y_n^{\sigma,O}]^\dagger \in \mathbb{R}^{n \times d_2}$, $\sigma \in \{a, b\}$. With (6), the cluster-level contrastive loss becomes

$$L_{C_2} \triangleq L_{\mathrm{C}}(\mathbf{c}^{a,O}, \mathbf{c}^{b,O}; \tau_C) + H(\mathbf{c}^{a,O}) + H(\mathbf{c}^{b,O}). \tag{17}$$

Hence, the final loss of NC+C is defined as

$$L_{\mathrm{NC+C}} = L_{\mathrm{NC}} + L_{\mathrm{C}_2}. \tag{18}$$

Non-Contrastive+Non-Contrastive (NC+NC): We utilize the non-contrastive (4) as the instance-level loss and the KL divergence (12) for clustering. Due to the structure of the two networks processing two augmented data in BYOL, we only consider the representation from the online network f^O. Therefore, given $\sigma \in \{a, b\}$, $q_{i,m}^{\sigma,O}$ is the probability that the representation $z_i^{\sigma,O}$ belongs to cluster m with center μ_m, and $p_{i,m}^{\sigma,O}$ defines the target distribution. The KL divergence loss in BYOL is

$$L_{\mathrm{NC}_3} \triangleq \frac{1}{N} \sum_i \Big[\mathrm{KL}(p_i^{a,O} || q_i^{a,O}) + \mathrm{KL}(p_i^{b,O} || q_i^{b,O})\Big]. \tag{19}$$

Hence, the final local loss for NC+NC is

$$L_{\mathrm{NC+NC}} = L_{\mathrm{NC}} + L_{\mathrm{NC}_3}. \tag{20}$$

B. Federated Extensions of the Two Stages

In FedSSRL, local training can be trained using either a single contrastive or a non-contrastive loss. This entails training each local model with a single SimCLR framework or a BYOL scheme in a FL setting, which are named as FedSimCLR and FedBYOL, respectively, in some literature [9]–[11]. In this paper, we adopt the notation 'FedC' to denote a single-stage FedSimCLR and 'FedNC' to denote a single-stage FedBYOL. TO perform federated clustering, a two-stage training process is necessary, comprising an instance-level representation stage followed by a clustering stage. Our idea is to extend the centralized loss functions (13), (16), (18) and (20) to the federated setting, resulting in our so-called models FedC+C, FedC+NC, FedNC+C, and FedNC+NC, respectively. Each of these four schemes is able to learn representations and generate clustering assignments jointly.

Let $\mathcal{D}_k \triangleq \{x_{1,k}, \ldots, x_{|\mathcal{D}_k|,k}\}$, $k = 1, \ldots, K$ represent the local datasets of the users. At Client k with local data $\mathcal{D}_k$, the local training optimizes the loss:

$$L_k \triangleq \sum_{x_{i,k} \in \mathcal{D}_k} L(x_{i,k}), \tag{21}$$

where $L(x_{i,k})$ can be any of the losses from L_{C} (2), L_{NC} (4), $L_{\mathrm{C+C}}$ (13), $L_{\mathrm{C+NC}}$ (16), $L_{\mathrm{NC+C}}$ (18), and $L_{\mathrm{NC+NC}}$ (20). Then, both FedSSRL and FedCl aim to optimize the objective

$$\min \sum_{k=1}^K \frac{|\mathcal{D}_k|}{|\mathcal{D}|} \sum_{x_{i,k} \in \mathcal{D}_k} L(x_{i,k}). \tag{22}$$

C. Empirical Study

To assess the performance of various loss functions L_k in federated learning, we implemented two evaluation tasks: a linear evaluation to evaluate the effectiveness of learned representation from FedSSRL, and a clustering analysis to determine the performance of FedCl. In the linear evaluation, we train a classifier on top of the frozen representations obtained from the global encoder. For the clustering task, we employ the global model's clustering assignments and evaluate the clustering accuracy. A more detailed description of our experimental setup can be found in Section VII.

The results on CIFAR-10 are summarized in Table I, with the best result highlighted in bold and the second-best in italics. In the linear evaluation process, we have the option of a single-stage representation learning utilizing either the non-contrastive loss or the contrastive loss. Alternatively, a two-stage training approach can be employed, with the first stage applying self-supervised learning loss and the second involving clustering loss. Table I indicates that representation learning based on non-contrastive loss surpasses methods based on contrastive loss in FL, with FedNC+NC yielding the highest accuracy, achieving 86.01% in IID settings and 83.46% in Non-IID scenarios. Meanwhile, a two-stage training approach using FedNC+NC outperforms a single-stage FedNC by margins of 1.72% for IID and 4.02% for Non-IID scenarios, respectively.

For clustering tasks, a two-stage training process is required to produce the clustering assignments. As shown in Table I, cluster-level contrastive loss is more effective compared to the non-contrastive loss during the clustering stage. Specifically, the FedC+C configuration attains the highest clustering accuracies, achieving 64.90% in IID scenarios and 24.35% in Non-IID scenarios. The FedNC+C method emerges as the second-best approach, yielding accuracies of 62.70% for IID and 23.89% for Non-IID, respectively. On the other hand, clustering loss based on non-contrastive results in poor performance in federated deep clustering tasks.

Our experiments in this section rely on very basic, but fundamental, base schemes for contrastive and non-contrastive learning. We have used these base schemes to optimize over the specific strategy of contrastive or non-contrastive learning for the specific task of FedSSRL or FedCl. In the next two sections, we introduce 1) The FedSSRL model, which leverages a clustering-guided federated representation learning (FedRLC) strategy to improve the quality of representations built upon FedNC+NC, and 2) the federated clustering with Past Negatives Pool (FedPNP) scheme, specifically designed to optimize federated deep clustering tasks based on FedC+C framework.

TABLE I
STUDY ON FEDSSRL AND FEDERATED CLUSTERING. IID & DATA-SPLIT NON-IID.

Dataset	**Linear Evaluation**		**Clustering Accuracy**	
Method	IID	Non-IID	IID	Non-IID
FedC	82.15	78.09	-	-
FedNC	*84.29*	*79.44*	-	-
FedC+NC	82.57	78.28	23.58	21.95
FedC+C	83.50	77.83	**64.90**	**24.35**
FedNC+NC	**86.01**	**83.46**	21.54	15.59
FedNC+C	83.90	78.84	*62.70*	*23.89*

V. FEDRLC: CLUSTERING-GUIDED FEDERATED LEARNING OF REPRESENTATIONS

A. Motivation

From the empirical study in Section IV, our findings indicate that optimizing cluster centers helps with learning representations. Thus, we develop FedRLC, which incorporates a crossed KL divergence loss with a data selection strategy during local training, and introduces a novel dynamic controller designed to update cluster centers during federated communication.

B. Local Training in FedRLC

The block diagram of the FedRLC framework is illustrated in Fig. 1 for local training at a certain Client k. In the following, we shall describe each stage in the figure in detail. The first stages to obtain the instance representations (until L^{INS}) apply verbatim from the BYOL scheme. We now describe the next steps.

In FedRLC, we define a novel crossed KL divergence loss (CKL) to learn a well-separated representation. CKL aims at optimizing M cluster centers by a crossed divergence between probabilities calculated from the online network and the target distribution from the target network. Specifically, given the input data $x_{i,k} \in \mathcal{D}_k$, $\alpha \in \{a, b\}$ and $\nu \in \{O, T\}$, the $q_{i,m,k}^{\alpha,\nu}$ and $p_{i,m,k}^{\alpha,\nu}$ is defined by (9) and (10). We can now compare the probabilities $q_{i,m,k}^{\alpha,O}$ induced by the online networks with the probabilities $p_{i,m,k}^{\alpha,T}$ of the target networks. By (11), the crossed KL divergence objective can be defined as

$$L^{\mathrm{CKL}_0} \triangleq \frac{1}{n} \sum_i \Big[\mathrm{KL}(p_{i,k}^{b,T} || q_{i,k}^{a,O}) + \mathrm{KL}(p_{i,k}^{a,T} || q_{i,k}^{b,O}) \Big], \quad (23)$$

where N represents the batch size. The crossed KL objective (23) intends to optimize the local cluster centers by incorporating information from both augmented views of the input. The two augmented samples are supposed to share similar probabilities because they are derived from the same data.

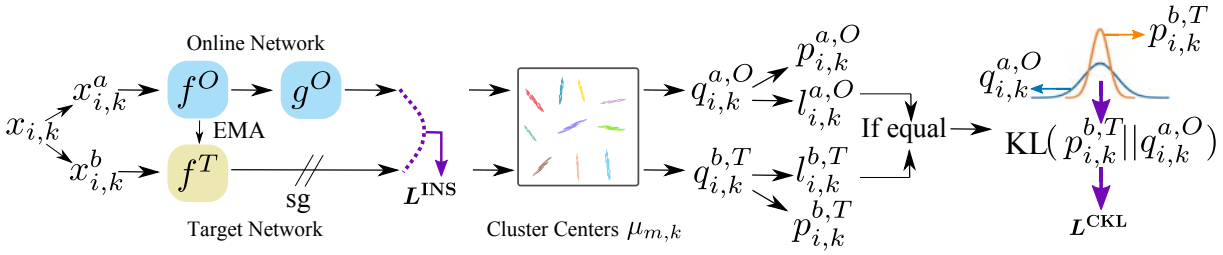


Fig. 1. The FedRLC framework during local training. sg means stop gradient. In the figure, we illustrate the construction of the first terms of the symmetric loss function in (24); the second terms are similar.

Another novelty that we incorporate in FedRLC is to make sure that the augmentations that are involved in the crossed KL objective in (23) are not too far. Indeed, intuitively, completely irrelevant augmentations would harm, instead of benefit the overall performance. This is why we only incorporate pairs whose hard decisions match in the KL divergence losses. Let $l_{i,k}^{\alpha,\nu} = \text{argmax}_m(q_{i,m,k}^{\alpha,\nu})$ denote the hard clustering decisions of the online and target networks with different augmentations. Ties are broken in favor of the smallest index.

The data is chosen to contribute to the crossed KL divergence loss only when the predicted label from the online and the target networks are the same. We thus modify the loss in (23) to work with

$$L^{\text{CKL}} \triangleq \frac{1}{n}\sum\{\text{KL}(p_{i,k}^{b,T}||q_{i,k}^{a,O}) : l_{i,k}^{b,T} = l_{i,k}^{a,O}\}+ \frac{1}{n}\sum\{\text{KL}(p_{i,k}^{a,T}||q_{i,k}^{b,O}) : l_{i,k}^{a,T} = l_{i,k}^{b,O}\}. \quad (24)$$

As shown in Figure 1, we jointly optimize the cluster centers and the online/target networks during local training. Therefore, the overall loss function is given by

$$L_{\text{FedRLC}} = L^{\text{CKL}} + L^{\text{INS}}, \quad (25)$$

where L^{INS} recalls the classical instance-level NC loss defined in (4). Usually, a hyperparameter can be incorporated to the loss function to control the relative weight of the losses L^{CKL} and L^{INS}. In our experiments, equal weights on the losses already provided a good performance. We thus leave a detailed study on hyperparameter tuning as future work.

C. Updates After Server-to-Client Communications

We describe the cluster center and online network update mechanisms during the server-to-client communications. We use subscript $\star$ to denote the global models, the subscript k to be the local model, and the superscript O to be the online networks. Let r represent the current training round. During the communication update, only the cluster centers and the online network are updated. We now introduce a novel rule to update the centers.

Specifically, given centers $\{\mu_{m,k}^{r-1} \in \mathbb{R}^d\}_{m=1}^M$ in local user k with local data $\mathcal{D}_k$ at round $r-1$, global centers $\mu_{m,\star}^r$ at round r, the centers of Client k at round r are updated according to

$$\mu_{m,k}^r = \frac{\epsilon_r}{1+\epsilon_r}\mu_{m,k}^{r-1} + \Big(1 - \frac{\epsilon_r}{1+\epsilon_r}\Big)\mu_{m,\star}^r, \quad (26)$$

where ϵ_r is updated progressively by the KL divergence between the probability generated from the local and global centers. Specifically, letting $f_\star^r$ and $f_k^{O,r-1}$ denote the global encoder in round r and the local encoder in round $r-1$ at Client k, respectively, we define $z_{\star,i,k} \triangleq \frac{1}{2}(f_\star^r(x_{i,k}^a)+f_\star^r(x_{i,k}^b))$, $z_{i,k} \triangleq \frac{1}{2}(f_k^{O,r-1}(x_{i,k}^a)+ f_k^{O,r-1}(x_{i,k}^b))$ as the mean representations of data $x_{i,k}$ under different augmentations and with global and local networks. We now evaluate the soft class probabilities for the data of Client k according to the global model at Round r as $q_{\star,i,m,k} \triangleq \Delta_m(z_{\star,i,k}, \{\mu_{n,\star}^r\}_{n=1}^M)$. Likewise, we can evaluate the class probabilities according the local model at Round $r-1$ as $q_{i,m,k} \triangleq \Delta_m(z_{i,k}, \{\mu_{n,k}^{r-1}\}_{n=1}^M)$. We can now compute the momentum parameter ϵ_r via

$$\epsilon_r = \frac{1}{|\mathcal{D}_k|}\sum_{i=1}^{|\mathcal{D}_k|}\text{KL}\big(\{q_{\star,i,m,k}\}_{m=1}^M \| \{q_{i,m,k}\}_{m=1}^M\big). \quad (27)$$

In order to keep the features extracted from local data, it is important to the preserve the local model especially when the distance of the local model to the global model is large. This was also observed in previous work [10], [11]. In FedRLC, when ϵ_r is large, the divergence between probabilities generated from global and local networks is large, so that the cluster centers inherit more local knowledge. Otherwise, a smaller ϵ_r gathers more information from global cluster centers. Figure 2 illustrates the rule for updating cluster centers during each communication round.

Finally, we discuss how to update the client online networks. We follow the EMA scheme [10]. Specifically, the online networks at Round r are updated as

$$(f_k^{O,r}, g_k^{O,r}) \leftarrow \gamma(f_k^{O,r-1}, g_k^{O,r-1}) + (1-\gamma)(f_\star^{O,r}, g_\star^{O,r}). \quad (28)$$

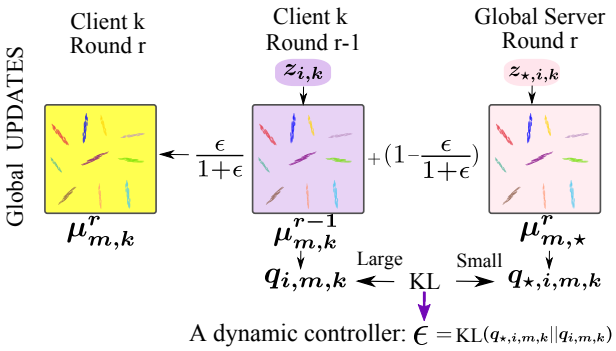


Fig. 2. Cluster centers update during communication round.

In (28), the parameter γ is used to control the weight between the global model and the local model. An explicit formula for γ is given by [10] $\gamma = \min(\lambda_k ||f_\star^r - f_k^{O,r-1}||, 1)$ where $\lambda_k = \frac{\tau}{||f_\star^1 - f_k^0||}$ is a customized magnitude, τ is a hyperparameter, and f is the encoder. In EMA [10], λ_k is only calculated once at the first round. Thus, ϵ_r is used to update the cluster centers $\mu_{m,k}^r$, while γ is applied to the update of the local online networks, including an encoder f^O and a predictor g^O. Algorithm 1 shows the overall FedRLC scheme.

Algorithm 1 FedRLC

Input: Number of communication rounds R, Number of clients K, Number of local epochs E.
Output: Global encoder $f_\star$ and predictor $g_\star$.

Server executes: Initialize server's network parameters $f_\star$, $g_\star$, and $\mu_{\star,m}$. Have the clients initialize local parameters f_k^O, g_k^O, and $\mu_{k,m}$
for $r = 1, \ldots, R$ **do**
  for $k = 1, 2, \ldots, K$ in parallel **do**
    Send global encoder $f_\star$, predictor $g_\star$, and cluster centers $\mu_{\star,m}$ to client k.
    $f_k^O, g_k^O, \mu_{m,k} \leftarrow$ **ClientTraining**$(f_\star, g_\star, \mu_{\star,m})$.
  end for
  FedAvg: $(f_\star^O, g_\star^O, \mu_{\star,m}) \leftarrow \sum_k \frac{|\mathcal{D}_k|}{|\mathcal{D}|}(f_k^O, g_k^O, \mu_{m,k})$.
end for
Return global encoder $f_\star$ and predictor $g_\star$.
ClientTraining$(f_k^O, g_k^O, \mu_{k,m})$
Update the online networks and cluster centers via global parameters by (28) and (26), respectively.
for epochs $= 1, \ldots, E$ and size-N batch learning within each epoch over dataset $\mathcal{D}_k$ **do**
  Update online networks and cluster centers via global parameters by descending the gradient of the local cost function in (25).
  Update the target network parameters f_k^T via (5).
end for
Return the online networks f_k^O and g_k^O.

VI. FedPNP: Federated Clustering with Past Negatives Pool

A. Motivation

As discussed in Section IV, the FedRLC does not perform well on clustering tasks, while the cluster-level contrastive loss based on C+C yields better clustering performance. Based on C+C, we propose a federated clustering framework with a novel past negatives pool (PNP) for intelligently selecting positive and negative samples for contrastive learning. PNP benefits FL and contrastive learning simultaneously, specifically, alleviating class collision for contrastive learning and reducing client-drift in FL.

B. FedPNP: Overall Framework

In FedPNP, we aggregate several local models trained in a fully unsupervised federated way to obtain a global model that outputs the cluster information directly. The overall block diagram of the FedPNP architecture is shown in Fig. 3. The central server contains global networks $f_\star, g_{I,\star}$, and $g_{C,\star}$, representing a base encoder, an instance-level projector, and a cluster-level projector, respectively. Let $f_k, g_{C,k}, g_{I,k}$ denote the network elements at user k. In each communication round, the central server sends global networks to local clients. Each local device updates the local model using its own local data and sends the updated model to the sever. The server updates the global networks by a weighted average of the local models. Finally, the clustering assignments can be obtained from the global cluster-level projector.

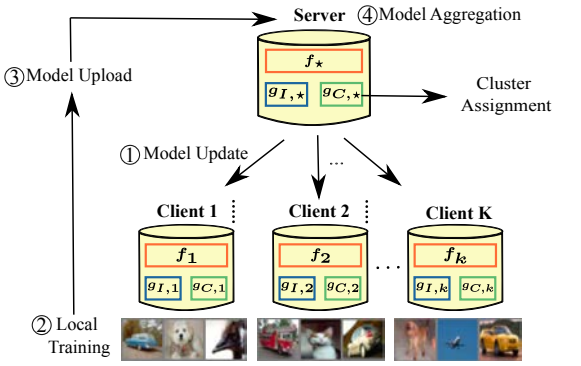


Fig. 3. The federated clustering scheme. At the client k, f_k, $g_{I,k}$, $g_{C,k}$ represent base encoder, instance-level projector, and cluster-level projector, respectively.

C. PNP and PNP loss

The proposed FedPNP relies on contrastive representation learning [5], [16], and specifically CC [16], as outlined in Section III. Simply extending the CC scheme to a federated setting results in poor performance we show in Section IV. This is because: 1) The Non-IID

data over multiple users causes the client-drift during local training. 2) FedSSRL needs to store more local information that may lose during fast aggregation in FL, which leads to poor representations. 3) Negative samples in contrastive loss causes class collision. In contrastive learning, the loss function (1) relies on a large number of so-called negative samples, which typically consist of all samples in a batch, excluding the input and its augmented version. The model is trained to distance these negatives far away from the input data. However, the negatives often contain data from the same category as the input, leading to what is known as class collision issue in contrastive learning.

We introduce the PNP for intelligently selecting negative samples in CL. Intuitively, in terms of the above issues 1) and 2), reducing client-drift contradicts with keeping more local data features. The PNP is designed to optimize the trade-off between these two demands. The idea is that we remove the potential positive samples by comparing soft labels produced from the past local models. We utilize features learned from the past to retain more local knowledge and avoid the client-drift during the current local training. Moreover, we compare the similarity between soft labels to select potential positives with the given input image and remove it from the large set of negatives in contrastive loss to alleviate class collision issue. Given data $x_{i,k} \in \mathcal{D}_k$, two augmented samples pass through not only local models in current communication but also through the local models in the previous round as shown in Fig. 4.

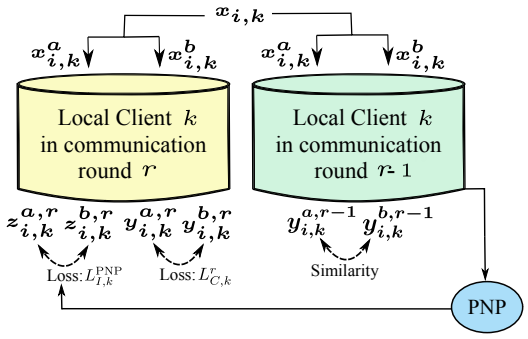


Fig. 4. The local training at client k for FedPNP

To construct the PNP, we compute the Gaussian similarity between the soft labels extracted from the past local model in communication round $r-1$. The Gaussian similarity measure is defined as $s(u,v) \triangleq \exp(u^\dagger v)$. The PNP for a given augmented data $x^a_{i,k}$ is created by

$$P_i = \{j \neq i : s(y^{a,r-1}_{i,k}, y^{\sigma,r-1}_{j,k}) < \mu,\ \forall \sigma \in \{a,b\}\}, \quad (29)$$

where μ is the threshold to select negative and positive samples, $y^{\sigma,r-1}_{i,k}$ represents the soft label extracted from the cluster-level MLP in the previous round $r-1$. Hence, sufficiently close samples form positive pairs while far samples are negatives. The key idea is that the indices P_i for negative pairs are obtained from the past model in FedPNP. This allows preserving local information, a key requirement in FedSSRL.

In particular, the decision whether a given sample is positive or negative is done according to the current model weights. To preserve more local information, we use the idea of PNP, and choose the negative samples over the indices described by the set (29) instead. With the PNP, we modify the traditional contrastive loss (1) to get PNP-contrastive loss

$$L^{\mathrm{PNP}}(\mathbf{u},\mathbf{v};\tau) \triangleq \frac{1}{n}\sum_{i=1}^{n} -\log \frac{s_\tau(u_i,v_i)}{\sum\limits_{j\in P_i}[s_\tau(u_i,u_j)+s_\tau(u_i,v_j)]}. \quad (30)$$

For FedSSRL, the PNP selects negatives from the past local models, which alleviates the client-drift during the current local update and maintains more local knowledge that is forgotten during the model aggregation. For contrastive representation learning, the PNP computes similarity between soft labels extracted from the past and removes samples that may have the same class category with the given input, which alleviates the class collision issue in traditional contrastive loss. In short, the PNP is beneficial in three aspects. 1) It avoids the large divergence of Non-IID networks updated locally in current communication round. 2) It keeps more local knowledge, which can be lost during model aggregation, benefiting FedSSRL. 3) It helps the class collision issue in traditional CL by removing potential positives from negative samples.

D. Local training in FedPNP

We now describe the training procedure at each client. We minimize the PNP-contrastive loss (30) on instance representation and ordinary contrastive loss (1) on both past and current cluster features. Formally, given a batch size n, the instance-level PNP-contrastive loss at user k is defined via the instance-level representations $\mathbf{z}^{\sigma,r}_k \triangleq [z^{\sigma,r}_{1,k} \cdots z^{\sigma,r}_{n,k}] \in \mathbb{R}^{d_1 \times n}$, $\sigma \in \{a,b\}$ as

$$L^{\mathrm{PNP}}_I \triangleq L^{\mathrm{PNP}}(\mathbf{z}^{a,r}_k, \mathbf{z}^{b,r}_k; \tau_I), \quad (31)$$

where $\tau_I > 0$ is the instance-level temperature.

On the other hand, given $\mathbf{c}^{\sigma,r}_k \triangleq [y^{\sigma,r}_{1,k} \cdots y^{\sigma,r}_{n,k}]^\dagger \in \mathbb{R}^{n\times d_2}$, $\sigma \in \{a,b\}$, we define the cluster-level contrastive loss at round r via (6) as

$$L^r_C \triangleq L_{\mathrm{C}}(\mathbf{c}^{a,r}_k, \mathbf{c}^{b,r}_k; \tau_C) + H(\mathbf{c}^{a,r}_k) + H(\mathbf{c}^{b,r}_k), \quad (32)$$

where $\tau_C > 0$ is the cluster-level temperature parameter. We combine the different performance measures described above into the overall loss function

$$L_{\mathrm{FedPNP}} \triangleq L^{\mathrm{PNP}}_I + \alpha L^r_C, \quad (33)$$

where α is the hyperparameter used to control the weight of the loss. The dependencies between the different losses are illustrated in Fig 4. Note that the first term in (33) depends on the parameters of both the current and the past network, while the second term depends only on the current parameters. To minimize the loss (33) in practice, we select the negatives from the current models only for $r = 1$ as there is no previous models in the first round. In the first round, the PNP is defined as $P_i = \{j \neq i : s(y_{i,k}^{a,r}, y_{j,k}^{\sigma,r}) < \mu,\ r = 1,\ \forall \sigma \in \{a, b\}\}$.

VII. Experiments

A. Experiment Setup

Evaluation Metrics: We divide our experiments into two parts: representation learning and clustering. We use benchmark datasets including CIFAR-10, CIFAR-100, where CIFAR10 has 10 classes, and CIFAR-100 has 100 classes. For CIFAR-10 and CIFAR-100, the training set contains 50,000 images and the testing set contains 10,000 data. For federated representation learning, we evaluate the learned representation from the global model using linear evaluation and semi-supervised learning, following the evaluation methods from recent FedSSRL works [9], [10]. For testing the clustering performance, we obtain the clustering assignment from the global model and evaluate it in terms of clustering accuracy (ACC), normalized mutual information (NMI), and adjusted rand index (ARI), which is also followed by the recent clustering works [16], [17], [42]. For a fair comparison, for all experiments trained in federated learning, we train the model for 100 communication rounds for $K = 5$ clients. For each communication round, each client is trained for $E = 5$ local epochs. We will make all computer codes for training and evaluation publicly available in the final version of the paper.

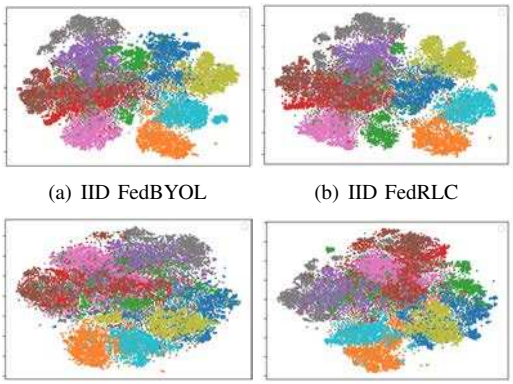
Data Spliting: We follow the same setup of FedU [9] and FedEMA [10], for a fair comparison. Namely, to simulate data heterogeneity in federated learning, each user only consists of samples from M/K classes, where M is the number of classes, and K is the number of clients. This is referred to as the data-split scenario. For independent and identically distributed (IID) data, each user has the same number of samples from M classes. In addition to the data-split non-IID scenario, to evaluate on different non-IID scenarios, we sample a specific proportion of the data from class m to client k, where the proportion is followed by the Dirichlet distribution with parameter β, which is also a widely-used method to simulate non-IID data distribution. A smaller β indicates a more heterogeneous distribution.

B. Representation Learning

Baselines: We evaluate FedRLC on linear evaluation and semi-supervised learning. Our baselines include FedU [9] and FedEMA [10], which are current state-of-the-art FedSSRL methods. We also evaluate FedBYOL, which refers to combining BYOL [6] with federated averaging. Single-Training refers to training each client independently, and the accuracy is calculated by the average of all clients. We also include results for the schemes discussed in Section IV, including FedSimDEC, FedCC, FedBYDEC, and FedBYCC.

Implementation Details: For FedSSRL, we adopt the SGD optimizer with a 0.032 initial learning rate. The learning rate is decayed by cosine annealing. The batch size is 128, and the input size is 32×32. We use ResNet18 to be the encoder, and the predictor is a two-layer multiplayer perceptron (MLP) with the output dimension 2048. The σ of EMA is 0.99, and the $\tau = 0.7$ is directly followed by [10] without tuning.

Visualization of Representations: To analyze the data features visually, we plot the t-SNE visualization of the CIFAR-10 learned from FedBYOL and FedRLC in Fig 5, where different colors indicate different classes. From the comparison between FedBYOL and FedRLC, we observe that the data representations obtained from FedRLC are separated more clearly. The following linear and semi-supervised evaluations further verify the effectiveness of FedRLC.



(a) IID FedBYOL (b) IID FedRLC

(c) Non-IID FedBYOL (d) Non-IID FedRLC

Fig. 5. t-SNE data visualization on CIFAR-10.

Linear Evaluation: To validate the quality of learned representations, a linear classifier is trained on top of the frozen representations learned from different FedSSRL methods. For linear evaluation training, the AdamW optimizer is adopted with a learning rate of 0.022. The results are shown in Table II and III. FedRLC constantly outperforms other methods, especially for CIFAR-100 with a large number of classes, where it improves by 2.77% and 1.62% on IID and non-IID data, respectively.

Semi-supervised Learning: We compare our model with state-of-the-art works on semi-supervised learning tasks. A new MLP is added on the top of the encoder

TABLE II
LINEAR EVALUATION: IID & DATA-SPLIT NON-IID.

Dataset	CIFAR-10		CIFAR-100	
Method	IID	Non-IID	IID	Non-IID
Single-Training	82.42	74.95	53.88	52.37
FedC+NC	82.57	78.28	55.31	54.43
FedC+C	83.50	77.83	52.79	51.28
FedNC+NC	86.01	83.46	60.09	61.30
FedNC+C	83.90	78.84	45.60	49.65
FedBYOL	84.29	79.44	54.24	57.51
FedU [9]	83.96	80.52	54.82	57.21
FedEMA [10]	86.26	83.34	58.55	61.78
FedRLC	**87.06**	**84.08**	**61.32**	**63.40**
BYOL (Centralized)	90.46		65.54	

TABLE III
LINEAR EVALUATION: DIRICHLET NON-IID.

Dataset	CIFAR-10		CIFAR-100	
β	0.5	0.1	0.5	0.1
Single-Training	83.42	83.08	58.45	57.20
FedC+NC	81.59	78.69	56.31	57.04
FedC+C	81.14	80.66	51.86	51.27
FedNC+NC	83.33	79.64	61.82	62.12
FedNC+C	83.07	81.14	46.62	47.90
FedBYOL	85.44	84.69	59.14	59.93
FedU	85.62	85.33	59.10	58.06
FedEMA	86.12	86.00	60.26	61.46
FedRLC	**86.89**	**86.69**	**62.39**	**63.21**

TABLE IV
SEMI-SUPERVISED LEARNING: IID & DATA-SPLIT NON-IID.

Dataset	CIFAR-10		CIFAR-100	
Method	IID	Non-IID	IID	Non-IID
Single-Training	78.08	69.06	43.50	39.99
FedC+NC	78.24	74.28	41.94	41.01
FedC+C	78.27	72.47	39.63	37.43
FedNC+NC	83.44	79.12	49.71	50.07
FedNC+C	79.92	71.20	32.38	35.87
FedBYOL	83.24	76.95	49.20	47.07
FedU	82.61	77.06	47.64	46.67
FedEMA	83.38	79.49	49.26	50.48
FedRLC	**83.99**	**79.52**	**49.67**	**52.16**

TABLE V
SEMI-SUPERVISED LEARNING: DIRICHLET NON-IID.

Dataset	CIFAR-10		CIFAR-100	
β	0.5	0.1	0.5	0.1
Single-Training	81.72	79.89	48.53	49.41
FedC+NC	76.93	75.29	42.90	43.16
FedC+C	76.34	74.68	38.86	37.80
FedNC+NC	82.82	81.18	49.80	49.86
FedNC+C	77.88	75.50	33.37	35.22
FedBYOL	82.84	82.20	50.00	50.12
FedU	81.33	81.66	49.25	49.31
FedEMA	83.18	82.06	50.11	51.07
FedRLC	**83.41**	**82.73**	**50.41**	**51.19**

in semi-supervised learning, and we fine-tune the entire model with 10% labeled data. We compare different federated representation learning methods under IID and Non-IID setting for CIFAR-10 and CIFAR-100 datasets. Tables IV and V demonstrate that our scheme achieves the best results in all cases. In particular, FedRLC improves the performance of CIFAR-100 by 1.68% under a highly heterogeneous scenario.

C. Clustering Results

Baselines: In Section IV, we have already presented several federated clustering results. Now, we will provide additional results for various data distributions and compare the proposed FedPNP with several existing SSRL methods in the context of federated clustering tasks. Also, we consider the centralized clustering scheme ProPos [42] as an upper bound. ProPos [42] only provides experiments for the 20 super-classes of the CIFAR-100 dataset while we consider the full 100 classes; we thus omit their results for a fair comparison.

Implementation Details: In this experiment, we use ResNet-18 [53] as the base encoder. We use Adam optimizer with an initial learning rate of 0.0003 and without weight decay. All input images are resized to 224×224, and the batch size n is set to 128. The output dimension of the instance-level MLP is set to 128, and the feature dimension of the cluster-level MLP is equal to the number of clusters. The instance-level temperature is $\tau_I = 0.5$, and the cluster-level temperature is $\tau_C = 1.0$. In FedPNP, μ is set to 0.999 for selecting negatives. We set the hyper-parameters $\alpha = 2$ for the first round $r = 1$ and $\alpha = 0.1$ starting from the second round. For 20 and 50 clients, we set $\alpha = 0.5$. For the Single-Training experiment, we train each client 300 epochs and report the mean clustering accuracy among all 5 clients by K-means. For all other federated clustering

methods, we show the performance based on the cluster assignment from the global cluster-level MLP or global cluster centers.

TABLE VI
CLUSTERING ACCURACY (%), IID DATA.

Dataset	CIFAR-10			CIFAR-100		
Method	NMI	ACC	ARI	NMI	ACC	ARI
Single-Training	45.7	56.1	33.1	34.1	16.3	8.6
FedC+NC	18.5	23.6	6.1	16.6	8.4	2.3
FedC+C	54.9	64.9	46.3	34.1	16.3	8.4
FedNC+NC	17.6	20.2	5.8	26.6	11.9	5.6
FedNC+C	53.8	62.7	44.9	33.2	15.7	7.9
FedPNP (ours)	**56.8**	**66.5**	**47.1**	**34.7**	**17.1**	**9.0**
ProPos [42] (Centralized)	88.6	94.3	88.4	-	-	-

TABLE VII
CLUSTERING ACCURACY (%), NON-IID DATA ($\beta = 0.5$).

Dataset	CIFAR-10			CIFAR-100		
Method	NMI	ACC	ARI	NMI	ACC	ARI
Single-Training	40.6	46.4	25.7	33.9	16.2	8.4
FedC+NC	18.4	20.6	5.9	16.5	7.8	2.2
FedC+C	41.2	44.7	27.5	34.4	16.5	8.9
FedNC+NC	18.3	20.1	5.5	26.1	11.5	4.4
FedNC+C	41.3	45.7	28.3	33.9	16.3	8.3
FedPNP (ours)	**42.7**	**49.5**	**30.5**	**34.5**	**17.0**	**8.9**
ProPos [42] (Centralized)	88.6	94.3	88.4	-	-	-

TABLE VIII
CLUSTERING ACCURACY (%), NON-IID DATA ($\beta = 0.1$).

Dataset	CIFAR-10			CIFAR-100		
Method	NMI	ACC	ARI	NMI	ACC	ARI
Single-Training	34.8	40.4	20.1	33.8	15.2	7.3
FedC+NC	15.4	25.4	8.5	15.3	6.9	2.0
FedC+C	33.2	38.0	20.3	33.0	15.0	7.3
FedNC+NC	13.2	21.5	7.1	27.7	12.3	5.7
FedNC+C	35.5	40.4	22.9	31.9	14.9	7.2
FedPNP (ours)	**36.4**	**43.7**	**24.2**	**34.1**	**16.0**	**7.9**
ProPos [42] (Centralized)	88.6	94.3	88.4	-	-	-

Federated Clustering Performance: Table VI shows the proposed FedPNP constantly outperforms other methods and achieves the best clustering performance in all data distribution settings. In IID cases, we improve the clustering accuracy by 10.4%, 3.7%, and 2% when comparing with baselines on CIFAR-10. Compared to simply doing a CC framework in Non-IID setting, we improve the clustering accuracy by 4.6% and 5.7 % for $\beta = 0.5$ and $\beta = 0.1$, respectively. For CIFAR-100 with 100 clusters, FedPNP is still the best approach for dealing with such large number of classes. FedPNP outperforms all other methods, achieving higher accuracy in fewer communication rounds. We note that fewer rounds translate to fewer amount of computations, which is an important gain for resource-limited edge devices. Another byproduct of fewer rounds is reduced communication latency, especially when the client-to-server communication rates are low [54], [55]. FedPNP thus offers significant advantages for both power and communication-limited edge devices.

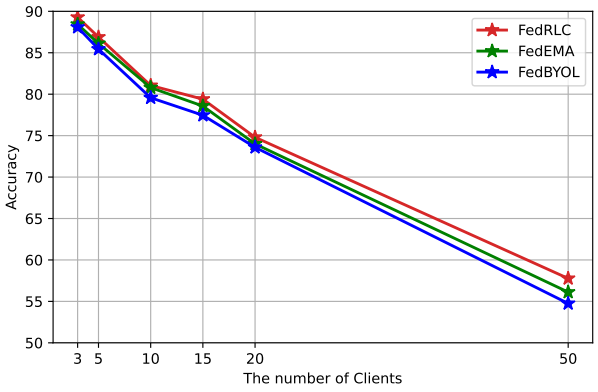


Fig. 6. Linear evaluation results on Non-IID CIFAR-10 under different numbers of clients.

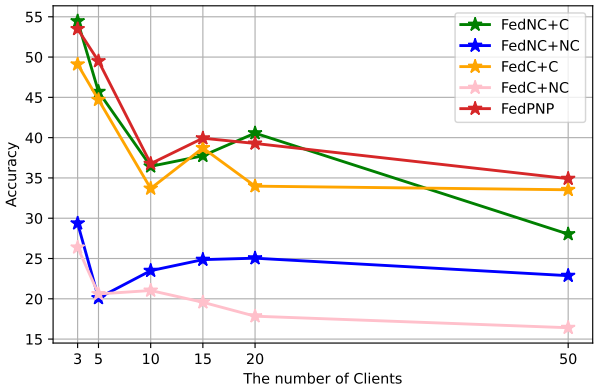


Fig. 7. Clustering performance on Non-IID CIFAR-10 under different numbers of clients.

VIII. ABLATION STUDIES

In this section, we perform ablation experiments over various factors, including varying the number of clients, simulating more Non-IID settings, the impact of data selection, and conducting experiments on the Tiny-ImageNet dataset to better understand the performance of FedRLC and FedPNP under different conditions.

Experiments on Varying the Number of Clients: We conduct the experiment with different numbers of nodes, as shown in Fig. 6 for FedRLC and Fig.7 for FedPNP. We report the linear evaluation accuracy for FedRLC and clustering accuracy for FedPNP. Compared to FedEMA

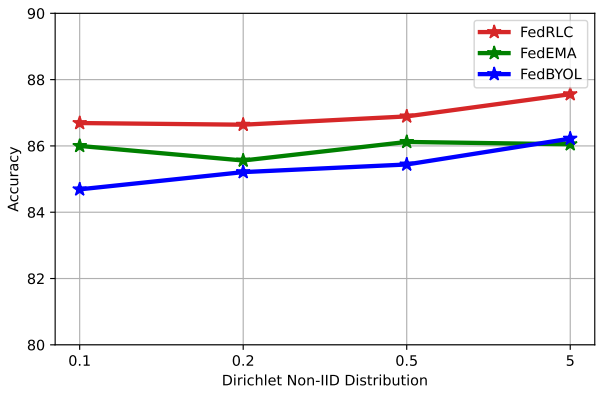


Fig. 8. Linear evaluation results on CIFAR10 with varying β of the Dirichlet Distribution.

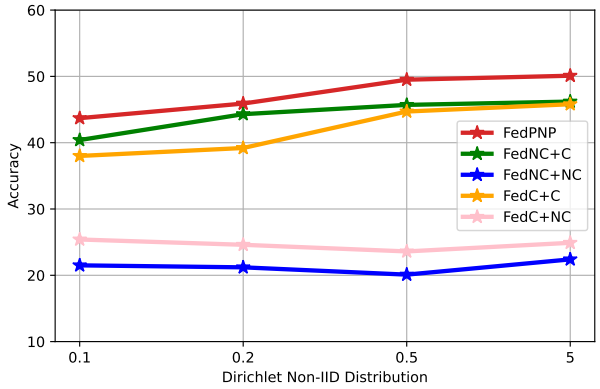


Fig. 9. Clustering performance on CIFAR10 with varying β of the Dirichlet Distribution.

and FedBYOL, we observe that all methods demonstrate a decline in accuracy as the number of clients increases, while FedRLC consistently performs the best among all methods.

In Fig. 6, compared to different combinations contrastive and non-contrastive strategies at the two stages of learning, we observe that FedRLC performs the best, especially when there are 50 clients. Also, as the number of clients increases, the performance gain also increases. Fig. 7 shows that our proposed federated clustering model, FedPNP, consistently performs well across different number of clients. Our model is either the best or within 1% of the best in terms of accuracy. Specifically, when there are 5 clients, FedPNP is over 5% higher in accuracy than the second-best model, FedNC+C. Moreover, FedPNP remains stable and robust as the number of clients changes, compared to other methods. In contrast, FedNC+C experiences a significant drop in accuracy of more than 10%, and FedCC is about 6% lower in accuracy than our model, FedPNP, when there are 20 clients.

Simulating More Non-IID Settings: In Fig. 8 and Fig. 9, we explore the effect of various Non-IID conditions by training the federated model with differing values of β in the Dirichlet distribution. A smaller β value indicates a more Non-IID situation. From Fig. 8, we find that FedRLC achieves the highest linear evaluation accuracy and maintains a relatively stable performance under different Non-IID conditions. Additionally, FedEMA outperforms FedBYOL but fluctuates slightly when varying the data distribution. Therefore, FedRLC demonstrates robustness in maintaining a relatively high linear evaluation accuracy despite changes in data distribution. The clustering performance under varying β values of the Dirichlet Non-IID data distribution for different federated clustering frameworks is shown in Fig. 9. FedPNP consistently outperforms all other methods across the range of β values, obtaining a 3.2% accuracy improvement over FedNC+C. Both FedC+NC and FedNC+NC show relatively low performance in all Non-IID scenarios, indicating the non-effectiveness of the non-contrastive method in federated clustering tasks.

Impact of Data Selection: In FedRLC, data selection is applied when the predicted labels from the online network and the target network are the same, as they are expected to output similar embeddings. The intuition behind this approach is to increase the likelihood of selecting data with more accurate soft labels for training. In FedPNP, we select data specifically for contrastive learning, which carries the challenging class collision problem. Class collision refers to the fact that contrastive learning sees all samples in a batch (excluding the input) as negatives to the input data, which is not always true, as some samples may belong to the same category as the input data. Existing works that address this issue primarily focus on centralized cases. However, FedPNP tackles the more challenging problem of class collision in the federated setting. We select data that are more likely to be a true negative for contrastive loss to alleviate the issue of class collision. More specifically, the data selection is based on Gaussian similarity between data embeddings generated from the previous network.

In TABLE IX, we compare the performance of FedRLC with and without data selection on CIFAR10 and CIFAR100 under different Non-IID data distributions, with $\beta = 0.5$ and under data-split Non-IID condition. We observe that the performance decreases after removing the data selection strategy, especially for the Non-IID CIFAR-100 dataset with a large number of classes, showing around a 2% drop in terms of linear evaluation accuracy. Additionally, removing the clustering loss proposed in FedRLC hurts the performance significantly, with a decrease of 4.64% and 5.89% for data-split Non-IID CIFAR-10 and CIFAR-100, respectively. Next, we evaluate the impact of data selection in FedPNP in TA-

BLE X. The model without the data selection performs worse than the proposed model in all settings in terms of federated clustering accuracy. Specifically, with a larger number of classes in CIFAR-100, the performance decreases by 1.5% in accuracy when the number of classes is 100 after removing the data selection strategy.

Impact of Hyperparameter: In TABLE X, we fine-tune the hyperparameter μ used for data selection in FedPNP with values of 0.998 and 0.996, compared to the original value of 0.999 in our setting shown in Section VII. The results show that FedPNP achieves optimal performance with $\mu = 0.999$ on Non-IID CIFAR-10 and $\mu = 0.996$ on IID CIFAR-10. For CIFAR-100, when $\mu = 0.999$, we obtain the best performance for Non-IID with 17.0%, and $\mu = 0.998$ for IID. Generally, we observe that the hyperparameter μ does not significantly impact performance, demonstrating that FedPNP is robust to variations in the hyperparameter when the data selection strategy is applied.

Experiments on Tiny-ImageNet for Transfer Learning: For large-scale datasets, we train the FedRLC on the Tiny-ImageNet dataset and fine-tune it on CIFAR-10 for linear evaluation. The results are presented in TABLE XI. Tiny-ImageNet comprises 200 classes and contains a total of $100,000$ samples, divided into 70% for the training set and 30% for the testing set. FedRLC achieves 41.4% linear evaluation accuracy, which is a 2% improvement compared to the model with some components removed. In all Non-IID scenarios, removing either the data selection or the clustering loss hurts the performance. These findings demonstrate that our data selection strategy is effective for large-scale data such as Tiny-ImageNet, even with its large number of classes, across various federated learning environments.

TABLE IX
EFFECT OF DATA SELECTION ON FEDRLC.

Dataset	CIFAR-10		CIFAR-100	
Method	Non-IID ($\beta = 0.5$)	Non-IID	Non-IID ($\beta = 0.5$)	Non-IID
FedRLC (No Data Selection)	86.01	83.51	60.82	63.03
FedRLC (Without Clustering Loss)	85.44	79.44	59.14	57.51
FedRLC	**86.89**	**84.08**	**62.39**	**63.40**

TABLE X
EFFECT OF DATA SELECTION ON FEDPNP.

Dataset	CIFAR-10		CIFAR-100	
Method	IID	Non-IID	IID	Non-IID
FedPNP (No Data Selection)	64.9	44.7	16.3	16.5
FedPNP ($\mu = 0.998$)	65.9	48.6	**17.6**	16.7
FedPNP ($\mu = 0.996$)	**66.7**	48.1	16.2	16.6
FedPNP	66.5	**49.5**	17.1	**17.0**

IX. CONCLUSIONS

We studied self-supervised representation learning and deep clustering algorithms in the federated setting. We introduced the FedRLC framework, designed to learn high-quality representations, and FedPNP, which automatically clusters data during training with non-IID unlabeled data. The experimental results demonstrated that FedRLC achieves state-of-the-art performance when evaluated through linear evaluation and semi-supervised learning. Additionally, FedPNP effectively generated cluster probabilities and outperformed many other methods in clustering data.

TABLE XI
EXPERIMENTAL RESULTS ON TRANSFER LEARNING.

Dataset	Tiny-ImageNet			
Method	IID	Non-IID	Non-IID ($\beta = 0.5$)	Non-IID ($\beta = 0.1$)
FedRLC (No Data Selection)	39.1	28.2	37.7	33.2
FedRLC (Without Clustering Loss)	40.4	28.5	38.8	32.6
FedRLC	**41.1**	**30.5**	**39.2**	**34.5**

REFERENCES

[1] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *ICML*, 2020.

[2] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., vol. 2, 2020, pp. 429–450.

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[4] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *CVPR*, June 2021, pp. 10 713–10 722.

[5] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th ICML*, 2020.

[6] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent - a new approach to self-supervised learning," in *NeurIPS*, 2020.

[7] X. Chen and K. He, "Exploring simple siamese representation learning," in *IEEE CVPR*, 2021.

[8] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, June 2020.

[9] W. Zhuang, X. Gan, Y. Wen, S. Zhang, and S. Yi, "Collaborative unsupervised visual representation learning from decentralized data," in *CVPR*, 2021, pp. 4912–4921.

[10] W. Zhuang, Y. Wen, and S. Zhang, "Divergence-aware federated self-supervised learning," in *ICLR*, 2022.

[11] L. Wang, K. Zhang, Y. Li, Y. Tian, and R. Tedrake, "Does learning from decentralized non-iid unlabeled data benefit from self supervision?" in *The Eleventh ICLR*, 2022.

[12] A. Zhong, H. He, Z. Ren, N. Li, and Q. Li, "Feddar: Federated domain-aware representation learning," in *Eleventh ICLR*, 2022.

[13] Y. Jin, Y. Liu, K. Chen, and Q. Yang, "Federated learning without full labels: A survey," *arXiv preprint arXiv:2303.14453*, 2023.

[14] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[15] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern recognition*, vol. 36, no. 2, pp. 451–461, 2003.

[16] Y. Li, P. Hu, J. Z. Liu, D. Peng, J. T. Zhou, and X. Peng, "Contrastive clustering," in *Thirty-Fifth AAAI*, 2021.

[17] Y. Li, M. Yang, D. Peng, T. Li, J. Huang, and X. Peng, "Twin contrastive learning for online clustering," *International Journal of Computer Vision*, vol. 130, no. 9, pp. 2205–2221, 2022.

[18] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *International conference on machine learning*. PMLR, 2016, pp. 478–487.

[19] H. Zhong, J. Wu, C. Chen, J. Huang, M. Deng, L. Nie, Z. Lin, and X.-S. Hua, "Graph contrastive clustering," in *CVPR*, 2021, pp. 9224–9233.

[20] S. Zhou, H. Xu, Z. Zheng, J. Chen, J. Bu, J. Wu, X. Wang, W. Zhu, M. Ester *et al.*, "A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions," *arXiv preprint arXiv:2206.07579*, 2022.

[21] L. Huang, A. L. Shea, H. Qian, A. Masurkar, H. Deng, and D. Liu, "Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records," *Journal of Biomedical Informatics*, vol. 99, p. 103291, 2019.

[22] S. Wang and T. Chang, "Federated clustering via matrix factorization models: From model averaging to gradient sharing," *CoRR*, vol. abs/2002.04930, 2020.

[23] I. S. Dhillon and D. S. Modha, "A data-clustering algorithm on distributed memory multiprocessors," in *Large-scale parallel data mining*. Springer, 2002, pp. 245–260.

[24] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," *arXiv:1203.6402*, 2012.

[25] T. Kucukyilmaz, U. of Turkish Aeronautical Association *et al.*, "Parallel k-means algorithm for shared memory multiprocessors," *J. Computer and Communications*, vol. 2, no. 11, p. 15, 2014.

[26] S. Yang, Z. Xie, H. Peng, M. Xu, M. Sun, and P. Li, "Dataset pruning: Reducing training data by examining generalization influence," in *The Eleventh International Conference on Learning Representations*, 2022.

[27] A. Gormez and E. Koyuncu, "Dataset pruning using early exit networks," in *ICML Workshop on Localized Learning (LLW)*, 2023.

[28] A. B. Ardic, H. Seferoglu, S. El Rouayheb, and E. Koyuncu, "Random walking snakes for decentralized learning at edge networks," in *IEEE LANMAN*, 2023.

[29] J. Parras and S. Zazo, "A graph network model for distributed learning with limited bandwidth links and privacy constraints," in *ICASSP*, 2020.

[30] R. Miao and E. Koyuncu, "Resource-efficient federated clustering with past negatives pool," in *IEEE ICASSP Worskhop on Timely and Private Machine Learning over Networks*, 2023, pp. 1–5.

[31] ——, "Federated representation learning through clustering," in *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*, 2023, pp. 1–6.

[32] ——, "Clustering-guided federated learning of representations," in *ICML Workshop on Fed. Learning and Analytics in Practice: Algorithms, Systems, Applications, and Opportunities*, 2023.

[33] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[34] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[35] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Neurips*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 3557–3568.

[36] J. Li, P. Zhou, C. Xiong, and S. C. Hoi, "Prototypical contrastive learning of unsupervised representations," *ICLR*, 2021.

[37] X. Wang, Z. Liu, and S. X. Yu, "Unsupervised feature learning by cross-level instance-group discrimination," in *CVPR*, 2021, pp. 12 586–12 595.

[38] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep adaptive image clustering," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5879–5887.

[39] J. Chang, G. Meng, L. Wang, S. Xiang, and C. Pan, "Deep self-evolution clustering," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 4, pp. 809–823, 2018.

[40] P. Haeusser, J. Plapp, V. Golkov, E. Aljalbout, and D. Cremers, "Associative deep clustering: Training a classification network with no labels," in *German Conf. Pattern Recognition*, 2018.

[41] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *ECCV*, 2018.

[42] Z. Huang, J. Chen, J. Zhang, and H. Shan, "Learning representation for clustering via prototype scattering and positive sampling," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2022.

[43] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *CVPR*, June 2020.

[44] Z. Huang, J. Chen, J. Zhang, and H. Shan, "Exploring non-contrastive representation learning for deep clustering," *abs/2111.11821*, 2021.

[45] M. Zheng, F. Wang, S. You, C. Qian, C. Zhang, X. Wang, and C. Xu, "Weakly supervised contrastive learning," in *ICCV*, October 2021, pp. 10 042–10 051.

[46] J. D. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka, "Contrastive learning with hard negative samples," in *ICLR*, 2021.

[47] Y. Kalantidis, M. B. Sariyildiz, N. Pion, P. Weinzaepfel, and D. Larlus, "Hard negative mixing for contrastive learning," in *NeurIPS*, vol. 33, 2020.

[48] H. Pan, D. Badawi, R. Miao, E. Koyuncu, and A. E. Cetin, "Multiplication-avoiding variant of power iteration with applications," in *ICASSP*, 2022, pp. 5608–5612.

[49] Y. Tian, H. Weng, and Y. Feng, "Towards the theory of unsupervised federated learning: Non-asymptotic analysis of federated em algorithms," in *Forty-first ICML*.

[50] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Neurips*, vol. 33, pp. 19 586–19 597, 2020.

[51] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *IJCNN*, 2020.

[52] Y. Mansour, M. Mohri, J. Ro, and A. T. Suresh, "Three approaches for personalization with applications to federated learning," *arXiv preprint arXiv:2002.10619*, 2020.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[54] P. Li, E. Koyuncu, and H. Seferoglu, "Respipe: Resilient model-distributed dnn training at edge networks," in *ICASSP*. IEEE, 2021, pp. 3660–3664.

[55] P. Li, H. Seferoglu, V. R. Dasari, and E. Koyuncu, "Model-distributed dnn training for memory-constrained edge computing devices," in *2021 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2021, pp. 1–6.

Runxuan Miao is a Ph.D. student studying at the University of Illinois Chicago, advised by Professor Erdem Koyuncu.

Erdem Koyuncu (SM'22) is an Associate Professor in the Department of Electrical and Computer Engineering at the University of Illinois Chicago.