Federated Momentum Contrastive Clustering

RUNXUAN MIAO, University Of Illinois Chicago, Chicago, USA
ERDEM KOYUNCU, University Of Illinois Chicago, Chicago, USA

Self-supervised representation learning and deep clustering are mutually beneficial to learn high-quality representations and cluster data simultaneously in centralized settings. However, it is not always feasible to gather large amounts of data at a central entity, considering data privacy requirements and computational resources. Federated Learning (FL) has been developed successfully to aggregate a global model while training on distributed local data, respecting the data privacy of edge devices. However, most FL research focuses on supervised learning algorithms. A fully unsupervised federated clustering scheme has not been considered in the existing literature. We present federated momentum contrastive clustering (FedMCC), a generic federated clustering framework that can not only cluster data automatically but also extract discriminative representations training from distributed local data over multiple users. In FedMCC, we demonstrate a two-stage federated learning paradigm where the first stage aims to learn differentiable instance embeddings and the second stage accounts for clustering data automatically. The experimental results show that FedMCC not only achieves superior clustering performance but also outperforms several existing federated self-supervised methods for linear evaluation and semi-supervised learning tasks. Additionally, FedMCC can easily be adapted to ordinary centralized clustering through what we call momentum contrastive clustering (MCC). We show that MCC achieves state-of-the-art clustering accuracy results in certain datasets such as STL-10 and ImageNet-10. We also present a method to reduce the memory footprint of our clustering schemes.

CCS Concepts: • Computing methodologies → Artificial intelligence; Machine learning; Computer vision representations.

Additional Key Words and Phrases: Federated learning, clustering, contrastive learning, unsupervised learning, representation learning.

1 INTRODUCTION

The number of resource-limited mobile and Internet of Things (IoT) devices have increased exponentially over the last decade [2]. With increasing demand for data privacy and prevalence of edge computing devices, Federated Learning (FL) has shown tremendous success in training distributed data while preserving user privacy [20, 28, 31]. Unfortunately, most work on FL focus on supervised learning. For example, FedProx [28] introduces a proximal term to reduce the large divergence caused by local update, and MOON [27] addresses the unbalanced data distribution problem via model-level contrastive learning to maximize the similarity between global and local models. These work require a lot of human labeled data for supervised learning, which costs high-level labor efforts. Therefore, it is necessary to train a generalized deep clustering model to reduce the expensive data labeling in FL.

Self-supervised or Unsupervised Representation Learning (RL) is a popular approach that learns remarkable instance representations while handling unlabeled data in various computer vision and natural language processing tasks [11, 38, 41]. Contrastive learning [6] (CL) and non-contrastive learning [13] are two main directions in RL to produce differentiable embeddings by comparing the similarity between instances. Recently, RL has been naturally combined with deep clustering approach, leading to a fashion way of extracting representations and generating
clustering assignment jointly. For example, GCC [43] constructs graphs to select positives and negatives for CL, and ProPos [19] updates the cluster prototypes to learn more compact cluster features. However, the utilization of these strategies presupposes centralized data collection, an approach that falls short in distributed data contexts, where concerns around privacy, limited computation capabilities, and resilience are critical [1, 15, 25, 32, 39]. Our objective is thus to develop an online clustering model through FL, which presents a more challenging but practical scenario when dealing with distributed data.

There are existing methods [42, 45, 46] exploring FL with unlabeled data, but they are primarily designed for linear evaluation or semi-supervised learning with fully or partially labeled data, thus not suitable for clustering tasks. For instance, FedU [45] adopts BYOL [13] framework, a representation learning technique based on mean-square error loss function and a momentum update, to federated learning and introduces a dynamic update between clients and the server for the predictor in BYOL. In these federated self-supervised learning work, they only output and evaluate the learned representations, thereby lack of the ability to solve clustering problems. Although a few algorithms are introduced for federated clustering, they are only effective on specific datasets such as medical datasets [18] and have difficulties with clustering complex data features [4, 10, 23, 24, 37]. For example, FedMAvg [37] uses classical K-means algorithm, but for clustering only simple synthetic datasets. Hence, training a more generic deep clustering model with FL is still an open research direction that has received little attention in the literature.

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.

For image data, it has been shown by many studies [14, 44] that deep clustering methods outperform traditional shallow clustering methods such as K-Means, spectral clustering, agglomerative clustering, and DBSCAN by leveraging neural networks to learn complex, hierarchical features. We are thus focusing on deep clustering in this paper. By filling the gap between federated learning and centralized deep clustering, we propose Federated Momentum Contrastive Clustering (FedMCC) framework, where a shared global model that aims at clustering data automatically as well as training with unlabeled local data distributed over multiple machines. In FedMCC, we first adopt the centralized clustering work, Contrastive Clustering [29] (CC) to federated learning. CC is a simple and effective deep clustering scheme that learns satisfactory representation and outputs clustering assignments simultaneously based on an instance-level and a cluster-level contrastive learning. Moreover, CC is able to effectively generate soft labels by the cluster-level projector whose output dimension matches the total number of clusters given by the dataset. We observe in Section 4 that naively extending the CC scheme to FL leads to sub-optimal clustering performance. To improve the performance, we propose a two-stage learning strategy to design the FL stages of FedMCC. The first FL stage conducts the instance-level contrastive learning for learning high-quality representations. The second FL stage conducts cluster-level contrastive learning on the frozen representations learned from the first stage for clustering. Finally, the clustering assignment can be obtained from the global model aggregated from the second FL stage.

For local updates, similar to BYOL [13], FedMCC produces symmetric representations for both instance-stage FL and cluster-stage FL from two neural networks: a trained online network and a target network with a slow-moving average of the online parameters. In FedMCC, clustering assignments can be obtained from the final global model without running an additional K-means step. The experimental results show that FedMCC achieves the best clustering performance compared with several baselines. We also evaluate the representation quality learned from the first FL stage by comparing with the existing federated unsupervised learning methods. The
results show that FedMCC is not only effective for solving federated clustering problems but also outperforms some state-of-the-art federated representation learning methods. To emphasize our points, the existing federated representation methods cannot cluster data, while FedMCC is designed specifically to do clustering. Additionally, we show that the proposed FedMCC can be easily adopted to the centralized case as well and achieves state-of-the-art results on benchmark datasets including STL-10 and ImageNet-10. We summarize the main contributions of our work as follows:

- To the best of our knowledge, the proposed FedMCC is the first general federated online deep clustering framework that clusters data automatically while training with distributed data. The FedMCC outperforms various baseline by comfortable margins on clustering on various benchmark datasets.
- Our two-stage FL scheme enables learning data representations and data clustering. While our goal is to train a global model for clustering, at the same time, the model learned from the first FL stage generates remarkable representations that is competitive with the existing work on federated self-supervised learning methods.
- FedMCC can even be adapted to ordinary centralized clustering. The resulting MCC scheme outperforms existing methods on STL-10 and ImageNet-10 datasets.
- We also describe an algorithm to reduce the memory footprint of our schemes.

Overall, our proposed FedMCC scheme is a simple and effective federated clustering framework for edge devices that can be tailored for various other learning tasks.

1.1 Organization

The rest of the paper is organized as follows: In Section 2, we start with background on CC block. In Section 3, we introduce the FedMCC, and its variant centralized MCC. Numerical experiments are provided in Section 4, and ablation studies are reported in Section 5. We discuss how to design memory-efficient schemes in Section 6. Finally, in Section 7, we draw our main conclusions.

2 PRELIMINARIES: CONTRASTIVE CLUSTERING (CC)

The block diagram of the CC scheme is shown in Fig. 1. We consider a dataset \(D = \{x_1, \ldots, x_{|D|}\}\). Given an input \(x_i \in D\), the CC scheme first creates two samples \(x_i^a = t^a(x_i)\) and \(x_i^b = t^b(x_i)\) through transformations \(t^a\) and \(t^b\). We use the variable \(\sigma \in \{a, b\}\) to represent the sample index so that the transformations are succinctly expressed as \(x_i^\sigma = t^\sigma(x_i), \sigma \in \{a, b\}\). 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 \(h_i^a = f(x_i^a), \sigma \in \{a, b\}\). An instance-level multi-layer perceptron (MLP) \(g_I\) projects \(h_i^a\) and \(h_i^b\) to obtain instance-level representations \(z_i^\sigma = g_I(h_i^\sigma) \in \mathbb{R}^{d_1}, \sigma \in \{a, b\}\). Likewise, a cluster-level MLP \(g_C\) produces cluster-level representations \(y_i^\sigma = g_C(h_i^\sigma) \in \mathbb{R}^{d_2}, \sigma \in \{a, b\}\). 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 \(d_2\).
The similarity of any two representations are compared via the cosine similarity measure $s(u, v) \triangleq u^\top v/(\|u\|\|v\|)$. To define the loss functions, we need the following definitions. Given matrices $u = [u_1 \cdots u_N] \in \mathbb{R}^{d \times n}$ and $v = [v_1 \cdots v_N] \in \mathbb{R}^{d \times n}$ constructed via the indicated column vectors, we define the contrastive loss function

$$L(u, v; \tau) \triangleq \frac{1}{n} \sum_{i=1}^{n} -\log \frac{\exp(\frac{1}{\tau} s(u_i, v_i))}{\sum_{j \neq i} \exp(\frac{1}{\tau} s(u_i, v_j)) + \exp(\frac{1}{\tau} s(u_i, v_i))}. \quad (1)$$

Given a batch size $n$, the instance-level contrastive loss is then defined via the instance-level representations $z^\sigma \triangleq [z_1^\sigma \cdots z_n^\sigma] \in \mathbb{R}^{d_1 \times n}$, $\sigma \in \{a, b\}$ as $L(z^a, z^b; \tau_1)$, where $\tau_1 > 0$ is the instance-level temperature parameter. On the other hand, given $c^\sigma \triangleq [c_1^\sigma \cdots c_n^\sigma] \in \mathbb{R}^{d_2 \times n}$, $\sigma \in \{a, b\}$, the cluster-level contrastive loss is defined by $L(c^a, c^b; \tau_C)$, where $\tau_C > 0$ is the cluster-level temperature. The cluster-level contrastive loss is utilized to differentiate clusters. Specifically, columns of $c^a$ and $c^b$ are considered as the representation of each cluster. On the other hand, rows of $z^a$ and $z^b$ (i.e. $y_n^\sigma$s) correspond to the soft labels of samples. In particular, in a deterministic assignment of inputs to clusters, all rows would be one-hot encoded vectors.

The overall loss function of CC is an entropy-regularized linear combination of instance-level and cluster-level losses. In precise form, the loss function is given by

$$L_{CC} \triangleq \frac{1}{2}L(z^a, z^b; \tau_1) + L(c^a, c^b; \tau_C) + H(c^a) + H(c^b), \quad (2)$$

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

$$H(u) \triangleq -\sum_{i=1}^{d} \frac{\|u_i\|_1}{\|u\|_1} \log \frac{|u_i|_1}{\|u\|_1}. \quad (3)$$

As discussed in [29], entropy regularization helps avoid the trivial solution where all samples are assigned to the same cluster.

Under the concerns of data privacy and limited computation resources, it is not practical to always assume that the data is centrally available. As a result, clients cannot send their local data to a central server to train a clustering model in federated learning. The resulting federated clustering framework is described next.

3 FEDMCC FRAMEWORK

3.1 Motivation and Problem Statement

We aim at addressing the above issue by building on the CC block with a novel multi-stage federated clustering scheme FedMCC. In FedMCC, we aggregate several local models trained in a fully unsupervised federated way to obtain a global model that outputs the cluster information directly. In particular, suppose there are $K$ clients, where client $k$ has its local unlabeled data $D_k$. Our goal is to learn a machine model over the dataset $D \triangleq \bigcup_{k=1}^{K} D_k$ on the central server that can cluster a group of unlabeled data automatically. A first idea for designing a federated clustering scheme is therefore to optimize the loss (2) at each user. However, it turns out that this strategy results in poor performance, as shown by the experimental results later. To improve the performance, we also replicate each block in the CC scheme to two different “online” and “target” networks. The parameters of the target network are calculated by using an exponential moving average (EMA) of the parameters of the online network. We refer to [13, 19] that utilize variants of this idea, however for the mean-squared error (MSE) loss function. The MSE loss provides poor performance in our settings. We shall thus utilize instead the loss in (1).

Cluster-level representations typically have much lower dimensionality than instance-level representations. We thus expect that training consistent cluster-level representations across multiple users with heterogeneous datasets is much more difficult than training consistent instance-level representations. Hence, a key idea of this paper is to advocate a multi-stage solution to the federated clustering problem. One should first construct consistent
high-dimensional representations across users, which can ultimately be reduced to cluster representations in a consistent manner. In the case of an extension of centralized clustering, this is done by breaking apart the centralized cost function into the two parts: instance-level contrastive loss and cluster-level contrastive loss described in the following part.

3.2 Two-stage FedMCC

In this section, we introduce FedMCC to solve the federated clustering problems with a novel multi-stage FL scheme. Let \( \mathcal{D}_k = \{x_{1,k}, \ldots, x_{|\mathcal{D}_k|, k}\} \) \( k = 1, \ldots, K \) represent the local datasets of the users. Given input \( x_{i,k} \in \mathcal{D} \) at user \( k \), we create the two samples \( t^\sigma(x_{i,k}) \), \( \sigma \in \{a, b\} \), similar to CC. The two samples are processed by what we call an “online network” and a “target network” simultaneously. We use the superscripts “\( O \)” and “\( T \)” to represent variables related to online and target networks, respectively. For each user \( k \), the online network consists of an encoder \( f^O_k \), an instance-level projector \( g^O_{I,k} \), and a cluster-level projector \( g^O_{C,k} \). The target network has the same architecture as the online network; consisting of an encoder \( f^T_k \), an instance-level projector \( g^T_{I,k} \), and a cluster-level projector \( g^T_{C,k} \). At each epoch and each batch update, only the parameters of the online networks are updated via gradient descent, and the target network parameters are kept frozen. The target network parameters are then updated via an EMA filter via \( \mathcal{P}(v^T) ← m\mathcal{P}(v^T) + (1-m)\mathcal{P}(v^O) \), \( v \in \{f, g^C, g^I\} \), where \( \mathcal{P}(\cdot) \) represents the parameter set of its argument. Also, the momentum factor or the decay rate \( m \in (0, 1) \) is a hyperparameter to be set and fixed before one initiates gradient descent updates.

![Fig. 2. The FedMCC scheme for the special case of two clients.](image)

To define the loss functions, for each user \( k \in \{1, \ldots, K\} \), \( \sigma \in \{a, b\} \) representing the sample index, and \( v \in \{O, T\} \) representing the network index, we define the representation vectors

\[
z_{i,k}^\sigma ≜ g^v_{i,k}(f^v_{i,k}(t^\sigma(x_{i,k}))), \quad g^\sigma_{i,k} ≜ g^v_{i,k}(f^v_{i,k}(t^\sigma(x_{i,k}))),
\]

and their matricized versions \( z_k^{\sigma,v} ≜ [z_{1,k}^{\sigma,v} \ldots z_{|\mathcal{D}_k|, k}^{\sigma,v}] \), and \( c_k^{\sigma,v} ≜ [y_{1,k}^{\sigma,v} \ldots y_{|\mathcal{D}_k|, k}^{\sigma,v}] \). We now define the instance-level contrastive loss of user \( k \) as

\[
L_{I,k} = \frac{1}{2}(L(z_k^{a,O}, z_k^{b,T}; \tau_I) + L(z_k^{a,T}, z_k^{b,O}; \tau_I)),
\]

(5)

and the cluster-level contrastive loss of user \( k \) as

\[
L_{C,k} = \frac{1}{2}(L(e_k^{a,O}, e_k^{b,T}; \tau_C) + L(e_k^{a,T}, e_k^{b,O}; \tau_C)) + H(e_k^{a,O}) + H(e_k^{b,O}) + H(e_k^{a,T}) + H(e_k^{b,T}).
\]

(6)

Corresponding to the two loss functions in (5) and (6), FedMCC relies on a two-stage learning scheme as shown in Fig. 2. In the first representation learning stage, the clients optimize the loss function (5), and the learned models are combined at the server every \( E \) epochs through federated averaging. Once the first stage converges, in the second clustering stage, the clients optimize the loss function in (6). A detailed algorithmic description of the two
Algorithm 1 FedMCC: Representation Learning

**Input:** Number of communication rounds $R$, Number of clients $K$, Number of local epochs $E$.

**Output:** Global encoder $f_*$

1. **Server executes:** Initialize server’s network parameters $f_*$ and $g_{l,*}$.
2. for $r = 1, \ldots, R$ do
   3. for $k = 1, 2, \ldots, K$ in parallel do
   4. Send global encoder $f_*$ and global instance-level projector $g_{l,*}$ to client $k$.
   5. $f_k, g_{l,k} \leftarrow \text{ClientLocalTraining}(k, f_*, g_{l,*})$
   6. end for
   7. Federated averaging: $P(f_*) \leftarrow \frac{1}{\sum_k |D_k|} \sum_k |D_k| P(f_k), P(g_{l,*}) \leftarrow \frac{1}{\sum_k |D_k|} \sum_k |D_k| P(g_{l,k})$.
8. end for
9. Return global encoder $f_*$.

Algorithm 2 FedMCC: Clustering

**Input:** Number of communication rounds $R$, Number of clients $K$, Number of local epochs $E$. A pre-trained (via Algorithm 1) global encoder $f_*$ available at each client $f_k = f_*, \forall k$.

**Output:** Global cluster projector $g_{C,*}$, cluster assignments.

1. **Server executes:** Initialize $g_{C,*}$.
2. for $r = 1, \ldots, R$ do
   3. for $k = 1, 2, \ldots, K$ in parallel do
   4. Send global cluster-level projector $g_{C,*}$ to client $k$.
   5. $g_{C,k} \leftarrow \text{ClientLocalTraining}(k, g_{C,*})$
   6. end for
   7. Federated averaging: $P(g_{C,*}) \leftarrow \frac{1}{\sum_k |D_k|} \sum_k |D_k| P(g_{C,k})$
8. end for
9. Return global cluster projector $g_{C,*}$.

**Test phase:** Compute the cluster assignment of a test image $x$ by $c \leftarrow \arg \max_x g_{C,*}(f_*(x))$.

11. **ClientLocalTraining**($k, g_{C,k}$)
12. for epochs $= 1, \ldots, E$ and size-$n$ batch learning within each epoch over dataset $D_k$ do
13. Update the online network $g_{C,k}^O$ of client $k$ by descending the gradient of the cluster-level contrastive loss $L_{C,k}$ in (6).
14. Update the target network $g_{C,k}^T$ of client $k$ via EMA.
15. end for
16. Return the online cluster-level projector $g_{C,k}^O$ of client $k$. 
We show the FedMCC can be easily adopted to centralized setting, resulting in MCC. Numerical results in Section 3 will show that the resulting MCC scheme not only achieves the best clustering accuracy performance available in the literature for certain datasets, but also improves the performance of CC for any given dataset. Our centralized scheme, MCC, is trained using all available data, similar to other centralized deep clustering frameworks. In this scenario, we assume that the data is centrally stored on a powerful server, which is reliable and will not fail. Consequently, this ensures that there will be no bottlenecks in communication and computation. If the server is not reliable or available, serverless decentralized learning schemes such as random walk learning [3] can be incorporated. In the centralized setting, similar to the notation in FedMCC without considering multiple users, given input \( x_i \in D \) from a central server, \( \sigma \in \{ a, b \} \) represents the sample index as before, and \( v \in \{ O, T \} \) represents the network index. The instance-level and cluster-level representations generated from the online and target networks are defined as

\[
z^{\sigma,v}_i \triangleq g^v_i(f^v(\sigma(x_i))), \quad y^{\sigma,v}_i \triangleq g^v_c(f^v(\sigma(x_i))), \quad \sigma \in \{a,b\}, \quad v \in \{O,T\}. \tag{7}
\]

To define the loss function, we collect the representation vectors into matrices

\[
z^{\sigma,v} \triangleq [z^{\sigma,v}_1 \cdots z^{\sigma,v}_n] \in \mathbb{R}^{d \times n}, \quad y^{\sigma,v} \triangleq [y^{\sigma,v}_1 \cdots y^{\sigma,v}_n]^T \in \mathbb{R}^{n \times d}, \quad \sigma \in \{a,b\}, \quad v \in \{O,T\}. \tag{8}
\]

Losses are evaluated across the outputs of the online and target networks for different augmentations and indicated by arrows in Fig. 3. As in the CC scheme, we apply entropy regularization to the cluster-level representations. Mathematically, for a given batch, the loss function for MCC is expressed as

\[
L_{MCC} \triangleq \frac{1}{2} (L(z^{a,O}, z^{b,T}; \tau_l) + L(z^{b,T}, z^{a,O}; \tau_l) + L(c^{a,O}, c^{b,T}; \tau_c) + L(c^{a,T}, c^{b,O}; \tau_c) + H(c^{a,O}) + H(c^{b,O}) + H(c^{a,T}) + H(c^{b,T}). \tag{9}
\]

The parameters of the target network are calculated by using an exponential moving average of the parameters of the online network. Different from FedMCC, instance-level and cluster-level losses are optimized together in FedMCC. Specifically, (9) represents the loss function used in our centralized scheme MCC, which is a linear combination of two contrastive losses: an instance-level contrastive loss and a cluster-level contrastive loss. In the proposed FedMCC scheme, we implement a two-stage federated learning approach for federated clustering, where we separate the instance-level and cluster-level contrastive losses. The loss function in (5) represents the instance-level contrastive loss used in the first stage, while (6) is the cluster-level contrastive loss employed in the second stage of clustering.

4 EXPERIMENTS

In this section, we present numerical results that demonstrate the performance of our federated and centralized clustering schemes over different datasets and scenarios. We first present the general experiment setup. We
then consider the clustering performance of FedMCC in the federated setting. We then evaluate the quality of representations generated by the first stage of FedMCC in the linear evaluation and semi-supervised learning settings. Finally, we present centralized clustering MCC results.

4.1 Experiment Setup

Datasets and Settings: We have conducted experiments on the CIFAR-10, CIFAR-100, STL-10, and Imagenet-10 datasets. The CIFAR-10 and CIFAR-100 datasets have 10 and 100 classes, respectively. All datasets consist of 50,000 training samples and 10,000 test samples with the same number of data samples per class. We have followed the same train-test split settings for STL-10 and ImageNet10, as in the recent clustering works [19, 29, 36]. Both datasets contain 13,000 samples and 10 classes. We show the performance of our schemes in terms of clustering accuracy (ACC), normalized mutual information (NMI), and adjusted rand index (ARI). Also, following the CC scheme [29], 100,000 unlabeled samples are trained in addition for instance-level representation learning for STL-10.

In the federated settings, for fair comparison with existing methods [42, 45, 46], we simulate a centralized node as the server and K distributed nodes as clients. In the IID scenario, each client contains the same number of images from all classes. In the non-IID scenario, we follow previous studies [42, 45, 46] to allocate the instances of class \(c\) to client \(k\) in a proportion of \(p_{jk}\) followed by the Dirichlet distribution \(\text{Dir}(\beta)\) (\(\beta = 0.5\) by default).

Implementation Details: For fair comparisons, we use ResNet-34 [17] as the backbone to report the centralized clustering results. In the federated setting, following the existing work [42, 45, 46], we use ResNet-18 or ResNet-50 [17] as encoders and train the model for 100 communication rounds for \(K = 5\) clients. For each communication round, each client is trained for \(E = 5\) local epochs. In the second stage, 100 communication rounds for cluster-level projector (Algorithm 2) is used for clustering, which is not considered for comparing the performance on linear evaluation and semi-supervised learning. The first stage of FedMCC comprises an encoder and a 2-layer MLP instance-level projector, while the second stage includes a 2-layer MLP serving as the cluster projector. In FedMCC,
the initialization of all model weights adheres to PyTorch’s default strategy for random initialization. For the particular neural networks in this study, this means utilizing a uniform distribution for all neurons. The range of the uniform distribution is determined in proportion to the inverse square root of the neuron’s in-degree.

The Adam optimizer [21] with an initial learning rate of 0.0003 and no weight decay is used. The input size of images is resized to 224×224 in both federated and centralized scenarios. The output dimension of the instance-level projector is set to 128, and the feature dimension of the cluster-level projector is equal to the number of clusters. Unless specified otherwise, we set the batch size to be $n = 128$, the instance-level temperature as $T_1 = 0.5$, the cluster-level temperature as $T_C = 1.0$, and the EMA parameter to be $m = 0.99$.

### 4.2 Federated Clustering Evaluation

**Baselines:** We consider clustering the representations generated by existing federated representation learning schemes and compare the resulting performance with that of FedMCC. In particular, we have implemented FedU [45], and clustered the resulting representations. We have considered FedU specifically as it does not require extensive hyperparameter tuning and performs very close to state of the art in linear evaluation and semi-supervised learning tasks, as we shall also show later. We have considered “FedU + K-means,” which refers to the combination of FedU with the classical K-means [30] algorithm. We also considered the baseline “FedU + MCC,” which refers to the combination of FedU with our second stage of federated clustering. We update FedU for 100 rounds and update 100 rounds of a cluster-level projector as we do in FedMCC. The baseline “FedCC” simply optimizes the CC loss 2 via federated averaging. FedMCC (Combined Loss) combines two stages and optimizes losses $L_{I,k} + L_{C,k}$ in (5) and (6) for 100 rounds, which is the same loss function as MCC in (9). As an upper bound, we have considered the centralized clustering scheme ProPos [19]. For CIFAR-100, we use 100 classes for federating setting and 20 super classes for centralized setting, which is followed by the existing works in centralized setting [19, 29] and federated setting [45].

| Dataset Method       | CIFAR-10 | CIFAR-100 | STL-10 | ImageNet-10 |
|----------------------|----------|-----------|--------|-------------|
|                      | NMI | ACC | ARI | NMI | ACC | ARI | NMI | ACC | ARI | NMI | ACC | ARI |
| FedU + K-means       | 47.0  | 57.6 | 36.1 | 37.0 | 21.9 | 11.5 | 35.2 | 39.4 | 20.2 | 49.9 | 61.8 | 37.8 |
| FedU + MCC           | 52.9  | 62.5 | 43.9 | 31.7 | 16.7 | 6.9  | 35.2 | 41.8 | 21.9 | 53.7 | 63.1 | 44.9 |
| FedCC                | 54.6  | 64.2 | 45.4 | 28.7 | 14.9 | 6.9  | 41.8 | 49.3 | 30.7 | 53.3 | 63.6 | 50.4 |
| FedMCC (Combined Loss) | 57.2  | 65.6 | 47.7 | 36.7 | 18.4 | 10.3 | 24.7 | 35.9 | 15.5 | 66.2 | 66.9 | 60.5 |
| FedMCC (ours)        | 57.0  | 69.4 | 50.6 | 35.6 | 22.0 | 11.8 | 41.7 | 49.6 | 30.8 | 61.6 | 67.3 | 54.1 |
| CC (Centralized)     | 70.5  | 79.0 | 63.7 | -    | -    | -    | 76.4 | 85.0 | 72.6 | 85.9 | 89.3 | 82.2 |
| MCC (Centralized)    | 76.2  | 84.9 | 72.7 | -    | -    | -    | 72.7 | 87.0 | 74.8 | 90.4 | 95.9 | 91.1 |
| ProPos [19] (Centralized) | 88.6  | 94.3 | 88.4 | -    | -    | -    | 75.8 | 86.7 | 73.7 | 89.6 | 95.6 | 90.6 |

**Results:** Tables 1 and 2 show the clustering accuracies and metrics of our FedMCC compared with baselines. We observe that FedMCC consistently outperforms all baselines. We observe that the naive federated generalization of the CC scheme, FedCC and the momentum version FedMCC (Combined Loss), are suboptimal in general. Specifically, for CIFAR-10, there is 5.2% and 6.1% improvement over FedCC for IID and Non-IID scenarios, respectively. Moreover, on the IID CIFAR-10 dataset, there is 6.9% improvement over FedU-based schemes. Note that FedU requires additional K-means for clustering, which is time consuming, especially for high-dimensional representations with large number of clusters such as CIFAR-100. We observe that the performance of FedMCC is

ACM Trans. Intell. Syst. Technol.
lower compared to the centralized scheme ProPos [19], which is considered an upper bound where all data are aggregated for training. We found that there is a notable big gap between the performance of FedMCC and the upper bounds, suggesting that there is potentially much room for improvement for the general task of federated clustering. For CIFAR-10 and IID data, we visualize the t-SNE embeddings produced by the cluster projector during the communication rounds in Fig. 4. FedMCC progressively trains the embedding from indistinguishable at the beginning to clear at the end.

### Table 2. Clustering accuracy (%) on non-IID datasets

| Dataset             | Method          | CIFAR-10 | CIFAR-100 | STL-10 | ImageNet-10 |
|---------------------|-----------------|----------|-----------|---------|-------------|
|                     |                 | NMI      | ACC       | ARI     | NMI         | ACC       | ARI     | NMI      | ACC       | ARI     |
| FedU + K-means      |                 | 45.8     | 48.7      | 30.0    | 21.9       | 11.5      | 34.8    | 38.9     | 19.9      | 40.0    |
| FedU + MCC          |                 | 38.2     | 47.2      | 26.6    | 31.7       | 16.7      | 6.9     | 33.6     | 19.6      | 39.8    |
| FedCC               |                 | 38.4     | 43.4      | 25.6    | 30.5       | 14.7      | 5.8     | 34.2     | 20.1      | 44.3    |
| FedMCC (Combined Loss) |             | 40.3     | 46.1      | 26.7    | 34.3       | 16.5      | 8.3     | 20.9     | 12.8      | 39.4    |
| FedMCC (ours)       |                 | 45.5     | 49.5      | 29.8    | 37.9       | 22.6      | 11.4    | 34.3     | 43.0      | 21.9    |
| CC (Centralized)    |                 | 70.5     | 79.0      | 63.7    | -          | -         | 76.4    | 85.0     | 72.6      | 85.9    |
| MCC (Centralized)   |                 | 76.2     | 84.9      | 72.7    | -          | -         | 72.7    | 87.0     | 74.8      | 90.4    |
| ProPos [19] (Centralized) |           | 88.6     | 94.3      | 88.4    | -          | -         | 75.8    | 86.7     | 73.7      | 89.6    |

### Table 3. Linear evaluation on Resnet-50

| Dataset | CIFAR-10 | CIFAR-100 |
|---------|----------|-----------|
| Method  | IID      | Non-IID   |
|---------|----------|-----------|
| Single client training [45] | 83.2 | 77.8 | 57.2 |
| FedSimCLR [6, 42] | 68.1 | 64.1 | 39.8 |
| FedCA [42] | 71.3 | 68.0 | 43.3 |
| FedSimSiam [8, 45] | 79.6 | 76.7 | 46.3 |
| FedU [45] | 86.5 | 83.2 | 59.5 |
| FedEMA [46] (λ = 0.8) | 86.1 | **85.3** | **60.9** |
| FedEMA [46] (τ = 0.7) | 85.1 | 84.3 | 59.5 |
| FedMCC | **86.8** | **85.3** | 60.8 |

### Table 4. Linear evaluation on Resnet-18

| Dataset | CIFAR-10 | CIFAR-100 |
|---------|----------|-----------|
| Method  | IID      | Non-IID   |
|---------|----------|-----------|
| Single client | 81.2 | 72.0 | 51.3 |
| FedU | 85.2 | 78.7 | 56.7 |
| FedEMA (λ = 0.8) | 85.6 | 82.8 | 57.9 |
| FedEMA (τ = 0.7) | 86.3 | 83.3 | 58.6 |
| FedMCC | 85.5 | 82.7 | 57.2 |
| FedMCC (Tuned) | **87.8** | **85.2** | **59.6** |

ACM Trans. Intell. Syst. Technol.
4.3 Linear and Semi-Supervised Learning Evaluation

Even though our focus in this paper is training a clustering scheme in the federated setting, we assess the performance of FedMCC on linear evaluation and semi-supervised setups to prove the effectiveness of the learned representations from the first stage of FedMCC.

**Linear Evaluation:** We train a linear classifier on top of the frozen representation trained by FedMCC and measure the Top 1% accuracy. Tables 3 and 4 compare our schemes with previous approaches [42, 45] by using different backbones. With a ResNet-50, FedMCC consistently outperforms FedEMA [46] all existing schemes on CIFAR10 dataset. The performance of FedMCC is still very close to FedEMA on CIFAR-100. Note that FedEMA utilizes mechanisms for tuning the parameters $\gamma$ and $\tau$ in the learning phase. If we also optimize $m$, we can push the accuracies further, as demonstrated in the results for ResNet-18. Although the untuned FedMCC with $m = 0.99$ falls short of FedEMA, by tuning $m = 0.988$, we can outperform FedEMA in most scenarios. In particular, with a ResNet-18, FedMCC obtains 85.2% top-1 accuracy on Non-IID CIFAR-10 dataset, which is almost 2% higher than the closest competitor.

**Semi-Supervised Learning:** We follow the same experimental procedure in [6, 40, 45] by using the labels of 1% or 10% of the samples during training on the CIFAR-10 dataset. Table 5 reveals that FedMCC improves over existing methods, especially for the non-IID setting, obtaining 82.0% top-1 accuracy for 10% labeled data with or without tuning the decay rate $m$. The tuned FedMCCs all use $m = 0.988$ as before.

4.4 Centralized Clustering Evaluation

We compare our MCC method with previous methods in Table 6. MCC achieves the state-of-the-art clustering accuracy on benchmark datasets including STL-10 with 87.0% and ImageNet-10 with 95.9%, showing the advantages of our proposed clustering scheme in centralized settings as well. Our MCC scheme also improves the performance of the CC scheme on all datasets. As compared with the best available methods, the performance of MCC falls short in certain datasets such as the ImageNet-Dogs and Tiny-ImageNet. MCC is still the second best for Tiny-ImageNet and the third best for ImageNet-Dogs among all methods. The underperformance of MCC on ImageNet-Dogs and Tiny ImageNet relative to the leading methods can be attributed to the high class count, where ImageNet-Dogs comprises 15 classes and Tiny-ImageNet encompasses 200 classes. In fact, we observe there is significant potential for improvement in the overall clustering performance across all methods on ImageNet-Dogs and Tiny ImageNet. Even the best clustering algorithms do not match the performance of our MCC.

![t-SNE plots of MCC for STL-10 dataset.](image-url)
not achieve high accuracy on these two datasets. Moreover, we note that CC achieves 42.9% and 14.0% clustering accuracy on ImageNet-Dogs and Tiny ImageNet, respectively. Our newly introduced approach, MCC, builds upon the foundation of CC, achieving a substantial enhancement with a 28% relative improvement on ImageNet-Dogs and a 9% increase on Tiny ImageNet, marking significant advancements. The specific nature of the datasets that makes MCC performs well (or worse) remains an interesting avenue for further research. We also provide the t-SNE plots on the STL-10 dataset in Fig. 5. Different colors denote the different predicted labels from the cluster-level projector. After 500 epochs, MCC produces clear boundaries for cluster assignments.

### Table 6. Clustering performance of different centralized algorithms

| Method      | NMI    | ACC    | ARI    | NMI    | ACC    | ARI    | NMI    | ACC    | ARI    | NMI    | ACC    | ARI    |
|-------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| k-means     | 12.5   | 19.2   | 6.1    | 11.9   | 24.1   | 5.7    | 5.5    | 10.5   | 2.0    | 6.5    | 2.5    | 0.5    |
| SC          | 9.8    | 15.9   | 4.8    | 15.1   | 27.4   | 7.6    | 3.8    | 11.1   | 1.3    | 6.3    | 2.2    | 0.4    |
| AE          | 25.0   | 30.3   | 16.1   | 21.0   | 31.7   | 15.2   | 10.4   | 18.5   | 7.3    | 6.9    | 2.7    | 0.5    |
| VAE         | 20.0   | 28.2   | 14.6   | 19.3   | 33.4   | 16.8   | 10.7   | 17.9   | 7.9    | 11.3   | 3.6    | 0.6    |
| JULE        | 18.2   | 27.7   | 16.4   | 17.5   | 30.0   | 13.0   | 5.4    | 13.8   | 2.8    | 10.2   | 3.3    | 0.6    |
| DEC         | 27.6   | 35.9   | 18.6   | 28.2   | 38.1   | 20.3   | 12.2   | 19.5   | 7.9    | 11.5   | 3.7    | 0.7    |
| DAC         | 36.6   | 47.0   | 25.7   | 39.4   | 52.7   | 30.2   | 21.9   | 27.5   | 11.1   | 19.0   | 6.6    | 1.7    |
| IIC         | 43.1   | 49.9   | 29.5   | -      | -      | -      | -      | -      | -      | -      | -      | -      |
| DCCM        | 37.6   | 48.2   | 26.2   | 60.8   | 71.0   | 55.5   | 32.1   | 38.3   | 18.2   | 22.4   | 10.8   | 3.8    |
| PICA        | -      | -      | -      | 78.2   | 85.0   | 73.3   | 33.6   | 32.4   | 17.9   | 27.7   | 9.8    | 4.0    |
| CC          | 76.4   | 85.0   | 72.6   | 85.9   | 89.3   | 82.2   | 44.5   | 42.9   | 27.4   | 34.0   | 14.0   | 7.1    |
| SCAN        | 80.9   | 69.8   | 64.6   | -      | -      | -      | -      | -      | -      | -      | -      | -      |
| GCC         | 68.4   | 78.8   | 63.1   | 84.2   | 90.1   | 82.2   | 49.0   | 52.6   | 36.2   | 34.7   | 13.8   | 7.5    |
| MiCE        | 63.5   | 75.2   | 57.5   | -      | -      | -      | 42.3   | 43.9   | 28.6   | -      | -      | -      |
| IDFID       | 64.3   | 75.6   | 57.5   | 89.8   | 95.4   | 90.1   | 54.6   | 59.1   | 41.3   | -      | -      | -      |
| PCL         | 71.8   | 41.0   | 67.0   | 84.1   | 90.7   | 82.2   | 44.0   | 41.2   | 29.9   | -      | -      | -      |
| ProPos [19] | 75.8   | 86.7   | 73.7   | 89.6   | 95.6   | 90.6   | **69.2** | **74.5** | **62.7** | **40.5** | **25.6** | **14.3** |
| MCC (ours)  | **77.2** | **87.0** | **74.8** | **90.4** | **95.9** | **91.1** | **53.0** | **54.7** | **39.5** | **35.09** | **15.2** | **7.5** |

### 5 ABLATION STUDIES

**Effect of Target Network:** To prove the effectiveness of the target network, we remove the exponential moving average of the online networks. Tables 1 and 2 show the performance of the resulting FedCC scheme, described in Section 4.2. Without the target networks, the performance decreases notably from FedMCC. A similar conclusion can be made from the results for the centralized scenario shown in Table 6, through observing the relative improvement of MCC over CC.

**Impact of Four Representations:** We recall that FedU and variants feed the two independent feature vectors to the online and target networks. On the other hand, FedMCC feeds vectors to both networks, resulting in four representations at both instance and cluster level. We can measure the resulting performance gains by comparing FedMCC with FedU + MCC. Table 6 reveals that the gains are within the range 6-7%.

**Target Network Aggregation:** We also conduct experiments by updating and aggregating both online and target networks during the communication round.
Table 7 shows the results when considering the target networks in communication protocol. We used the backbone ResNet-18 for this ablation study. Compared to the results in Tables 1 and 2, there is a major loss in performance in all scenarios.

**Effect of Local Epochs:** In Table 8, we consider 500 total epochs for each client. We vary local epochs $E = 1, 2, 5, 10$ with the corresponding number of communication rounds $R = 500, 250, 100, 50$. We can observe that both the number of local epochs and communication rounds are important, and there is a non-trivial optimal operating point.

**Impact of Momentum Parameter:** We have already seen through the results in Table 4 and 5 that optimizing the momentum parameter $m$ can improve the performance. Here, we show results for the choice $m = 0.996$ in Table 9, considered in the original BYOL paper [13]. The performance becomes noticeably worse, suggesting a strong dependence of the performance on $m$. Optimization of the decay rate for BYOL-type algorithms in the federated setting remains a very interesting avenue for further research. We have generally not performed decay rate optimization for fair comparison with existing studies (most previous work such as FedU also consider $m = 0.99$).

We now provide a more detailed technical justification to enhance the understanding of the original exponential moving averaging (EMA) strategy employed between two networks. During the training, only the online network is updated by gradient descent, while the target network is updated by EMA. The momentum parameter satisfies $m \in [0, 1]$.

The specific EMA mechanism of FedMCC allows retention of more past information at target networks, leading to more stable training. Stable training is particularly crucial in federated learning, given that large divergence between the global model and local updates leads to the client drift issue. On the other hand, rapid weighted averaging will result in the loss of local information, and is also not desirable. Hence, there is a fine balance between the global and local models in the sense that the local models should not remain too far as well as too close to the global model. Since EMA preserves more local information, it also helps to avoid fast aggregation caused by weighted averaging. Another notable advantage of EMA is that the augmented samples produced by the momentum network are more stable, effectively avoiding the class collision issue commonly encountered in contrastive learning. Therefore, the momentum network is beneficial in federated self-supervised representation learning.

### Table 7. Updating online & target nets

| Method        | Update Both |
|---------------|-------------|
| Dataset       | NMI | ACC | ARI | NMI | ACC | ARI | NMI | ACC | ARI |
| CIFAR-10 (IID) | 53.2 | 58.3 | 43.5 | 34.9 | 40.4 | 22.8 | 33.0 | 18.3 | 7.6 |
| CIFAR-10 (Non-IID) | 34.9 | 40.4 | 22.8 | 33.0 | 18.3 | 7.6 |
| CIFAR-100 (IID) | 30.9 | 15.5 | 6.4 |
| CIFAR-100 (Non-IID) | 30.9 | 15.5 | 6.4 |

### Table 8. Effect of local epochs

| Dataset        | CIFAR-10 (IID) | CIFAR-10 (Non-IID) | CIFAR-100 (IID) | CIFAR-100 (Non-IID) |
|----------------|---------------|--------------------|----------------|--------------------|
| Method         | NMI | ACC | ARI | NMI | ACC | ARI | NMI | ACC | ARI | NMI | ACC | ARI |
| $E = 1, R = 500$ | 48.7 | 57.6 | 36.2 | 39.5 | 40.9 | 25.9 | 34.6 | 19.2 | 8.4 | 32.3 | 17.1 | 7.1 |
| $E = 2, R = 250$ | 50.5 | 59.1 | 39.3 | 41.4 | 45.0 | 28.4 | 34.4 | 19.5 | 8.2 | 32.9 | 17.6 | 3.4 |
| $E = 5, R = 100$ | 57.0 | 69.5 | 45.0 | 45.5 | 49.5 | 29.8 | 35.6 | 22.0 | 11.8 | 37.9 | 22.6 | 11.4 |
| $E = 10, R = 50$ | 50.7 | 58.7 | 41.3 | 37.4 | 43.1 | 24.2 | 34.3 | 19.0 | 8.3 | 24.1 | 12.1 | 4.2 |

### Table 9. Effect of the momentum parameter

| Dataset        | CIFAR-10 (IID) | CIFAR-10 (Non-IID) | CIFAR-100 (IID) | CIFAR-100 (Non-IID) |
|----------------|---------------|--------------------|----------------|--------------------|
| Method         | NMI | ACC | ARI | NMI | ACC | ARI | NMI | ACC | ARI | NMI | ACC | ARI |
| FedMCC ($m = 0.996$) | 45.6 | 52.4 | 34.0 | 37.5 | 45.2 | 25.7 | 33.9 | 19.0 | 7.8 | 31.8 | 16.6 | 6.6 |
| FedMCC ($m = 0.99$) | 57.0 | 69.4 | 50.6 | 45.5 | 49.5 | 29.8 | 35.6 | 22.0 | 11.8 | 37.9 | 22.6 | 11.4 |
In the existing literature, numerous momentum-based self-supervised learning methodologies are studied, including BYOL [13], MoCo 1 and 2 [7, 16], and DINO [5]. These approaches have demonstrated superior performance utilizing the EMA strategy. Furthermore, the study in [35] has conducted extensive experiments to substantiate the hypothesis that momentum facilitates the encoder in learning from past data, thereby stabilizing the training process. Formal theoretical justifications as to why EMA or general momentum methods are helpful are largely missing in the literature, even in the centralized settings. We believe studying this problem in the centralized or more complex federated unsupervised learning setup is a fundamental and significant research direction that remains beyond the scope of the present work.

Communication Overhead: First, we note that in our numerical simulations, different schemes had been compared under the assumption of the same communication overhead, which stem from passing the model parameters between the server and the clients. In general federated learning, a global model is distributed by the server to several clients, who then update this model with their local training before returning it to the server. This process is a fundamental aspect of all federated learning frameworks. Our FedMCC approach does not incur any extra communication costs. Specifically, the models exchanged between the server and the clients include an ResNet encoder and a 2-layer MLP instance projector in the first stage of FedMCC. These components are consistent with those used in traditional self-supervised representation learning (SSL) frameworks, as introduced by the SimCLR approach [6], without adding any parameters that would increase communication load. Furthermore, the exchange of model parameters between the server and clients is crucial for implementing self-supervised representation learning within a federated context (FedSSL). Consequently, FedMCC avoids imposing additional communication overhead.

6 MEMORY-EFFICIENT MCC AND FEDMCC

An important special case of the federated learning paradigm is when the clients are low-cost edge devices that may have memory limitations [22, 26]. The amount of training memory needed for a naive implementation of the gradient descent updates for the loss function in (9) or its federated counterparts in (5) and (6) is linear in the batch size $n$. For the example case of the CC, this is because the network outputs $\{z_i^{\text{enc}}, y_i^{\text{proj}}\}, i = 1, \ldots, n$ for different members of a batch $x_i, i = 1, \ldots, n$ are coupled in a non-linear manner in the loss functions. Hence, to perform a gradient update, the size of the computation graph should be roughly of size $n\eta$, where $\eta$ represents the neural network size. This is in contrast to ordinary batch loss of the form $\ell(x_i)$, where $x_i$ are inputs and $\ell$ is some loss function. In this case, the gradient of the entire loss function can be calculated through accumulation using only one computation graph of size $\eta$.

Our idea for constructing memory-efficient MCC and FedMCC is akin to the scheme presented in [12] for the case of ordinary contrastive learning. We first calculate the gradients of the loss function with respect to the network outputs or representations. These gradients are calculated over one pass over the batch and stored on the memory. The memory overhead of this first stage is relatively low as the dimensionality of representation vectors is much smaller than that of the parameter space. A second pass over the batch can then update the network parameters. To describe the details of this procedure, let us now extend the CC scheme to a memory-efficient CC scheme. Extensions to MCC and FedMCC can be accomplished in the same fashion.
6.1 Construction of Memory-Efficient CC

We recall the loss function for the CC scheme in (2). The goal is to calculate the gradient $\frac{\partial L_{CC}}{\partial \theta}$, where $\theta$ represents the neural network parameters. For each $i \in \{1, \ldots, n\}$, let us define the gradients

$$
\begin{align*}
\alpha_{i,1} &\doteq \frac{1}{2} \frac{\partial L(x_i, z^{b}; \tau)}{\partial x_i^a}, \\
\alpha_{i,2} &\doteq \frac{1}{2} \frac{\partial L(x_i, z^{b}; \tau)}{\partial z^{b}_i}, \\
\alpha_{i,3} &\doteq \frac{1}{2} \frac{\partial L(c_i, c^{b}; \tau_C)}{\partial c_i^a}, \\
\alpha_{i,4} &\doteq \frac{1}{2} \frac{\partial L(c_i, c^{b}; \tau_C)}{\partial c^{b}_i}, \\
\alpha_{i,5} &\doteq \frac{1}{2} \frac{\partial H(c_i^a)}{\partial y^a_i}, \\
\alpha_{i,6} &\doteq \frac{1}{2} \frac{\partial H(c^{b}_i)}{\partial y^{b}_i}.
\end{align*}
$$

The partial derivative of the overall loss function with respect to one of the parameters $\theta^j \in \theta$ is then expressed as

$$
\frac{\partial L_{CC}}{\partial \theta^j} = \sum_{i=1}^{n} \left[ \left( \alpha_{i,1} \frac{\partial z^{a}_i}{\partial \theta^j} \right) + \left( \alpha_{i,2} \frac{\partial z^{b}_i}{\partial \theta^j} \right) + \left( \alpha_{i,3} + \alpha_{i,5} \right) \frac{\partial y^a_i}{\partial \theta^j} + \left( \alpha_{i,4} + \alpha_{i,6} \right) \frac{\partial y^{b}_i}{\partial \theta^j} \right].
$$

(12)

Note that, for any input index $i \in \{1, \ldots, n\}$ in a batch, the gradients $\frac{\partial z^{a}_i}{\partial \theta^j}, \frac{\partial z^{b}_i}{\partial \theta^j}, i \in \{a, b\}$ in (12) depend only on $x_i$ (and are independent of $x_j, j \neq i$). Hence, if $\alpha_{i,j}, i \in \{1, \ldots, n\}, j \in \{1, \ldots, 6\}$ are known, or calculated beforehand, the entire partial derivative (12) assumes the form of an “ordinary batch loss” $\sum_{i=1}^{n} \ell(x_i)$, as described in the beginning of this section. In fact, the parameters $\alpha_{i,j}$ can simply be calculated by a single forward pass of the entire batch (possibly one input at a time to save memory) before the gradient updates. As a result, once $\alpha_{i,j}$ are known, (12) can be calculated one input at a time through gradient accumulation. Moreover, for each given input, the standard backpropagation algorithm can be used to calculate the partial derivatives for all parameters of the network simultaneously.

Closed-form expressions for the gradients $\alpha_{i,j}, i \in \{1, \ldots, n\}, j \in \{1, \ldots, 6\}$ follow from cumbersome but basic calculus. To provide the final expressions, for any $i \in \{1, \ldots, n\}$, let

$$
\xi_i \doteq \sum_{j \neq i}^{n} \left[ \exp\left( \frac{1}{\xi}(s(u_i, u_j)) + \exp\left( \frac{1}{\xi}s(u_i, v_j)) \right) \right]
$$

(13)

We also define

$$
\begin{align*}
\frac{\partial s(u, v)}{\partial u} &\doteq \frac{s(u, v)}{\|u\|} \left( \frac{1}{\|v\|} u + \frac{1}{\|v\|} v \right),
\end{align*}
$$

(14)

using which we can provide the closed-form expressions

$$
\begin{align*}
n \tau \frac{\partial L(u,v; \tau)}{\partial u} &= -s'(u_i, u_\ell) + \sum_{i=1}^{n} \left[ \frac{1}{\xi}(s(u_i, u_\ell)) + \frac{1}{\xi}(s(v_i, u_\ell)) \right] + \exp\left( \frac{1}{\xi}s(u_\ell, u_i) \right) s'(u_\ell, u_i),
\end{align*}
$$

(15)

and

$$
\begin{align*}
n \tau \frac{\partial L(u, v; \tau)}{\partial v} &= -s'(v_\ell, u_\ell) + \sum_{i=1}^{n} \left[ \frac{1}{\xi}(s(u_i, u_\ell)) + \frac{1}{\xi}(s(v_i, u_\ell)) \right] s'(u_\ell, u_i),
\end{align*}
$$

(16)

for the gradients $L(u, v; \tau)$. In addition, with regards to the entropies that appear in the loss function, we can evaluate, for any $\ell \in \{1, \ldots, d\}$,

$$
\frac{\partial H(u)}{\partial u_\ell} = \sum_{i=1}^{d} \left[ \frac{1}{\|u_i\|_1 - 1(i = \ell)} \|u_i\|_1 \right] \left( 1 + \log \frac{\|u_i\|_1}{\|u_i\|_1} \right) \text{sign}(u_\ell).
$$

(17)
As mentioned, formulae (14)-(17) can be verified using straightforward calculations. Now, $\alpha_{i,1}$ and $\alpha_{i,2}$ in (10) can be calculated via (15). For $\alpha_{i,3}$, we can first use the identity

$$
\left[ \alpha_{1,3} \cdots \alpha_{n,3} \right] = \frac{1}{2} \left[ \frac{\partial L(e^a, c^b; \tau_C)}{\partial e^a_1} \cdots \frac{\partial L(e^a, c^b; \tau_C)}{\partial e^a_{\alpha_1}} \right]^\top
$$

where $e^a_1, \ldots, e^a_{\alpha_1}$ represent the columns of $e^a$. Now, the right hand side of (18) can be calculated through (16). Likewise, $\alpha_{i,4}, \alpha_{i,5}$, and $\alpha_{i,6}$ can be evaluated via (15), (16), (17), and the transposition method in (18).

The memory required for ordinary CC and our memory-efficient implementation of CC can be calculated to be $O((n_p + n_r)d)$ and $O(n_r n)$, respectively, where $n_r$ is the dimension of the representation space, and $n_p$ is the dimension of the parameter space. Since $n_r \ll n_p$ in practice, we expect significant savings in memory without sacrifice in performance. This will be corroborated via experiments in the next section.

6.2 Memory Consumption Experiments

In this section, we present experiments to validate our memory-efficient approach presented in Section 6.1. We implement both the standard gradient update and the memory-efficient version, reporting the GPU usage in Table 10 and Fig. 6 across various batch sizes. We consider the case of 5 users, the ResNet 18 model, and a 2-layer MLP as specified in Section 4. Table 10 details the memory usage during the local update stage of FedMCC, and Fig. 6(a) shows the comparison between the two methods. From Table 10, we observe that the standard method consumes a large amount of memory, particularly with larger batch sizes. In contrast, our method’s memory consumption remains relatively constant and is substantially lower than that of the standard update. As mentioned at the end of Section 6.1, the slight increase of our method’s memory consumption stems from the required memory to accommodate the batch representations. Still, it is important to emphasize that this quantity is relatively minor in comparison to ordinary CC method’s requirement to store entire neural network parameters, which requires huge amounts of memory. Fig. 6(a) further demonstrates that our approach maintains a near-constant and notably lower memory usage compared to the standard update, which increases linearly with batch size. Fig. 6(b) shows the clustering performance on CIFAR-10 Non-IID scenarios. As the computed gradients are identical, the accuracy obtained is the same for both methods, as shown in Fig. 6(b).

| Methods / Batch Size | 64   | 128  | 256   | 512  |
|----------------------|------|------|-------|------|
| Standard             | 5.7GB| 9.1GB| 16.0GB| 29.4GB|
| Memory-Efficient     | 2.6GB| 2.7GB| 3.0GB | 3.6GB|

Table 10. Memory cost comparison.

Our memory-efficient solution is mathematically equivalent to the standard computation. Therefore, we have reduced the memory consumption of the contrastive loss from $O((n_p + n_r)d)$ to $O(n_r n)$, where $n_r \ll n_p$, without incurring any loss in accuracy. Typically, the batch size $n$ is a very large number, especially in large language model such as transformers, where memory usage increases linearly with the batch size. For example, many vision-language models that are based on contrastive learning, such as CLIP [34] with a batch size of 32,768 and BASIC [33] with a batch size of 65,536, consume a significant amount of GPU resources. Many existing works [9] have also conducted extensive experiments demonstrating that a large batch size is beneficial to contrastive learning. Hence, memory-efficient MCC and FedMCC are important and can be applied to other general large language model learning methodologies based on contrastive learning.

ACM Trans. Intell. Syst. Technol.
Federated Momentum Contrastive Clustering

6 Fig. 6. Memory usage and the clustering accuracy over different batch sizes.

7 CONCLUSIONS

We have presented Federated Momentum Contrastive Clustering (FedMCC), a deep federated clustering model based on contrastive learning. FedMCC clusters a set of data points distributed over multiple clients and is trained by contrastive loss on a momentum network. Our method can be viewed as a fully unsupervised federated clustering scheme, extending existing research that primarily deals with self-supervised or semi-supervised representation learning. Our framework provides a distributed way to learn representations specialized for clustering. The extensive experiments demonstrate that the FedMCC provides good performance on the federated clustering scenario and achieves state-of-the-art clustering accuracy on several centralized datasets. In addition, FedMCC can also be adapted to linear evaluation and semi-supervised settings, achieving state-of-the-art results as well. Our approach opens up future interesting research directions such as clustering data with a large amount of clusters in the federated setting or improving the overall clustering accuracy.

ACKNOWLEDGMENTS

This work was supported in part by Army Research Lab (ARL) under Grant W911NF-21-2-0272, and by National Science Foundation (NSF) under Grant CNS-2148182.

REFERENCES

[1] Ahmed M Abdelmoniem, Atal Narayan Sahu, Marco Canini, and Suhaib A Fahmy. 2023. REFL: Resource-Efficient Federated Learning. In Proceedings of the Eighteenth European Conference on Computer Systems. 215–232.

[2] Mohammed H Alsharif, Jeong Kim, and Jin Hong Kim. 2017. Green and sustainable cellular base stations: An overview and future research directions. Energies 10, 5 (2017), 587.

[3] Alp Berke Ardic, Hulya Seferoglu, Salim El Rouayheb, and Erdem Koyuncu. 2023. Random Walking Snakes for Decentralized Learning at Edge Networks. In 2023 IEEE 29th International Symposium on Local and Metropolitan Area Networks (LANMAN). 1–6. https://doi.org/10.1109/LANMAN58293.2023.10189426

[4] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. 2012. Scalable K-Means++. arXiv:1203.6402 [cs,DB]

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In Proceedings of the IEEE/CVF international conference on computer vision. 9650–9660.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119), Hal Daumé III and Aarti Singh (Eds.). PMLR, 1597–1607. https://proceedings.mlr.press/v119/chen20j.html
[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. 2020. Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020).

[8] Xinlei Chen and Kaiming He. 2021. Exploring Simple Siamese Representation Learning. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021. Computer Vision Foundation / IEEE, 15750–15758. https://openaccess. thecvf.com/content/CVPR2021/html/Chen_Exploring_Simple_Siamese_Representation_Learning_CVPR_2021_paper.html

[9] Yihao Chen, Xianbiao Qi, Jianan Wang, and Lei Zhang. 2023. DisCo-CLIP: A Distributed Contrastive Loss for Memory Efficient CLIP Training. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 22648–22657.

[10] Inderjit S Dhillon and Dharmendra S Modha. 2002. A data-clustering algorithm on distributed memory multiprocessors. In Large-scale parallel data mining. Springer, 245–260.

[11] Chaowei Fang, Liang Wang, Dingwen Zhang, Jun Xu, Yixuan Yuan, and Junwei Han. 2022. Incremental cross-view mutual distillation for self-supervised medical CT synthesis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 20677–20686.

[12] Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021. Scaling deep contrastive learning batch size under memory limited setup. arXiv preprint arXiv:2101.06983 (2021).

[13] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap your own latent: A new approach to self-supervised learning. arXiv:2006.07733 [cs.LG]

[14] Xifeng Guo, En Zhu, Xinwang Liu, and Jianping Yin. 2018. Deep embedded clustering with data augmentation. In Asian conference on machine learning. PMLR, 550–557.

[15] Alperen Görmez, Venkat R. Dasari, and Erdem Koyuncu. 2022. E2CM: Early Exit via Class Means for Efficient Supervised and Unsupervised Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[16] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778. https://doi.org/10.1109/CVPR.2016.90

[18] Li Huang, Andrew L. Shea, Haining Qian, Aditya Masurkar, Hao Deng, and Jianbo Liu. 2019. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. Journal of Biomedical Informatics 99 (2019), 103291. https://doi.org/10.1016/j.jbi.2019.103291

[19] Zhizhong Huang, Ji Chen, Junping Zhang, and Hongming Shan. 2021. Exploring Non-Contrastive Representation Learning for Deep Clustering. CoRR abs/2111.11821 (2021). arXiv:2111.11821 https://arxiv.org/abs/2111.11821

[20] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119), Hal Daumé III and Aarti Singh (Eds.). PMLR, 5132–5143. https://proceedings.mlr.press/v119/karimireddy20a.html

[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980 [cs.LG]

[22] Erdem Koyuncu. 2023. Memorization Capacity of Neural Networks with Conditional Computation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 202677–202686.

[23] Erdem Koyuncu. 2024. Centroidal Clustering of Noisy Observations by Using rth Power Distortion Measures. IEEE Transactions on Neural Networks and Learning Systems 35, 1 (2024), 1430–1438. https://doi.org/10.1109/TNNLS.2022.3183294

[24] Tayfun Kucukyilmaz, University of Turkish Aeronautical Association, et al. 2014. Parallel k-means algorithm for shared memory multiprocessors. Journal of Computer and Communications 2, 11 (2014), 15.

[25] Pengzhen Li, Erdem Koyuncu, and Hulya Seferoglu. 2022. Adaptive and Resilient Model-Distributed Inference in Edge Computing Systems. IEEE Open Journal of the Communications Society 4 (2023), 1263–1273. https://doi.org/10.1109/OJCOMS.2023.3280174

[26] Pengzhen Li, Hulya Seferoglu, Venkat R Dasari, and Erdem Koyuncu. 2021. Model-distributed dnn training for memory-constrained edge computing devices. In 2021 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN). IEEE, 1–6.

[27] Qinbin Li, Bingsheng He, and Dawn Song. 2020. Federated Optimization in the Power 4 (2021). arXiv:2111.11821 https://arxiv.org/abs/2111.11821

[28] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated Optimization in Heterogeneous Networks. In Proceedings of Machine Learning and Systems, I. Dhillon, D. Papailiopoulos, and V. Sze (Eds.), Vol. 2, 429–450. https://proceedings.mlsys.org/paper2020/file/38af6f613bb65bf020ef33d30dd76442e-Paper.pdf

[29] Yunfan Li, Peng Hu, Jerry Zhaoli Liu, Dezhiou Peng, Joey Tianyi Zhou, and Xi Peng. 2021. Contrastive Clustering. In Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021. AAAI Press, ACM Trans. Intell. Syst. Technol.
[30] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (1, 14). Oakland, CA, USA, 281–297.

[31] Brendan McMaham, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguey y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54), Aarti Singh and Jerry Zhu (Eds.). PMLR, 1273–1282. https://proceedings.mlr.press/v54/mcmaham17a.html

[32] Hongyi Pan, Diaa Badawi, Runxuan Miao, Erdem Koyuncu, and Ahmet Enis Cetin. 2022. Multiplication-avoiding variant of power iteration with applications. In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 5608–5612.

[33] Hieu Pham, Zihang Dai, Golnaz Ghiasi, Kenji Kawaguchi, Hanxiao Liu, Adams Wei Yu, Jiahui Yu, Yi-Ting Chen, Minh-Thang Luong, Yonghui Wu, et al. 2021. Combined scaling for open-vocabulary image classification. arXiv e-prints (2021), arXiv:2111.

[34] Hieu Pham, Zihang Dai, Golnaz Ghiasi, Kenji Kawaguchi, Hanxiao Liu, Adams Wei Yu, Jiahui Yu, Yi-Ting Chen, Minh-Thang Luong, Yonghui Wu, et al. 2023. Combined scaling for zero-shot transfer learning. Neurocomputing 555 (2023), 126658.

[35] Trung Pham, Chaoming Zhang, Axi Niu, Kang Zhang, and Chang D Yoo. 2022. On the pros and cons of momentum encoder in self-supervised visual representation learning. arXiv preprint arXiv:2208.05744 (2022).

[36] Yuming Shen, Ziyi Shen, Menghan Wang, Jie Qin, Philip H. S. Torr, and Ling Shao. 2021. You Never Cluster Alone. CoRR abs/2106.01908 (2021). arXiv:2106.01908 https://arxiv.org/abs/2106.01908

[37] Shuai Wang and Tsung-Hui Chang. 2020. Federated Clustering via Matrix Factorization Models: From Model Averaging to Gradient Sharing. CoRR abs/2002.04930 (2020). arXiv:2002.04930 https://arxiv.org/abs/2002.04930

[38] Yang Yang, Bo Wang, Dingwen Zhang, Yixuan Yuan, Qingsen Yan, Shijie Zhao, Zheng You, and Junwei Han. 2023. Self-Supervised Interactive Embedding for One-Shot Organ Segmentation. IEEE Transactions on Biomedical Engineering (2023).

[39] H. Zhong, J. Wu, C. Chen, J. Huang, M. Deng, L. Nie, Z. Lin, and X. Hua. 2021. Graph Contrastive Clustering. In 2021 IEEE/CVF International Conference on Computer Vision (ICCV). https://doi.org/10.1109/ICCV48922.2021.00909

[40] Weiming Zhuang, Xin Gan, Yonggang Wen, Shuai Zhang, and Shuai Yi. 2021. Collaborative Unsupervised Visual Representation Learning from Decentralized Data. CoRR abs/2108.06492 (2021). arXiv:2108.06492 https://arxiv.org/abs/2108.06492

[41] Weiming Zhuang, Yonggang Wen, and Shuai Zhang. 2022. Divergence-aware Federated Self-Supervised Learning. In International Conference on Learning Representations. https://openreview.net/forum?id=oVE1z8NiNe