future internet

MDPI

Article

FedCO: Communication-Efficient Federated Learning via Clustering Optimization †

Ahmed A. Al-Saedi [1,*], Veselka Boeva [1] and Emiliano Casalicchio [1,2,*]

[1] Department of Computer Science, Blekinge Institute of Technology, SE-371 79 Karlskrona, Sweden
[2] Department of Computer Science, Sapienza University of Rome, 00185 Rome, Italy
* Correspondence: ahmed.a.al-saedi@bth.se (A.A.A.-S.); emiliano.casalicchio@uniroma1.it (E.C.)

† This paper is an extended version of our paper "Reducing Communication Overhead of Federated Learning through Clustering Analysis" published in Processing of the 2021 IEEE Symposium on Computers and Communications (ISCC), Athens, Greece, 5–8 September 2021.

Abstract: Federated Learning (FL) provides a promising solution for preserving privacy in learning shared models on distributed devices without sharing local data on a central server. However, most existing work shows that FL incurs high communication costs. To address this challenge, we propose a clustering-based federated solution, entitled Federated Learning via Clustering Optimization (FedCO), which optimizes model aggregation and reduces communication costs. In order to reduce the communication costs, we first divide the participating workers into groups based on the similarity of their model parameters and then select only one representative, the best performing worker, from each group to communicate with the central server. Then, in each successive round, we apply the Silhouette validation technique to check whether each representative is still made tight with its current cluster. If not, the representative is either moved into a more appropriate cluster or forms a cluster singleton. Finally, we use split optimization to update and improve the whole clustering solution. The updated clustering is used to select new cluster representatives. In that way, the proposed FedCO approach updates clusters by repeatedly evaluating and splitting clusters if doing so is necessary to improve the workers' partitioning. The potential of the proposed method is demonstrated on publicly available datasets and LEAF datasets under the IID and Non-IID data distribution settings. The experimental results indicate that our proposed FedCO approach is superior to the state-of-the-art FL approaches, i.e., FedAvg, FedProx, and CMFL, in reducing communication costs and achieving a better accuracy in both the IID and Non-IID cases.

Keywords: federated learning; Internet of Things; clustering; communication efficiency; convolutional neural network

Citation: Al-Saedi, A.A.; Boeva, V.; Casalicchio, E. FedCO: Communication-Efficient Federated Learning via Clustering Optimization. *Future Internet* **2022**, *14*, 377. https://doi.org/10.3390/fi14120377

Academic Editors: Qiang Duan and Zhihu Lu

Received: 4 November 2022
Accepted: 8 December 2022
Published: 13 December 2022

1. Introduction

With recent advances in Internet of Things (IoT) devices and the fast growth of high-speed networks, the need to collect and process vast amounts of distributed data generated by these devices is significantly increasing. Furthermore, Artificial Intelligence (AI) has concurrently transformed the discovery of knowledge methods with cutting-edge success in several applications, including text prediction, facial recognition, natural language processing, document identification, and other tasks [1,2]. However, those applications require IoT devices to send sensitive information to a remote cloud server for centralized model training, which raises data privacy concerns [3,4]. These privacy concerns of IoT devices are supposed to be reduced by introducing an alternative setting, i.e., Federated Learning (FL). The main idea of FL is to collaboratively train a shared machine learning model across distributed devices, where the data are stored locally on devices [5,6]. However, a naive implementation of the FL setting requires that each participant has to upload a full model update to a central server during each iteration. For large updates with millions of parameters for deep learning models and thousands of iterations [7], this step is likely

to be a major hindrance in FL when the network bandwidth is limited. Thus, Federated Learning can become completely impractical [8].

Over the past few years, there has been a growing consensus that the more data that can be guaranteed, the better and higher accuracy that will be achieved. It should not be assumed, however, that blindly introducing more data into a model will improve its accuracy, but only that ensuring high-quality data will guarantee a higher degree of accuracy.

Our Contributions: In this paper, we propose a novel FL framework, entitled Federated Learning via Clustering Optimization (FedCO), to lessen the challenges described above during the training process. In particular, FedCO draws inspiration from our previous work, Cluster Analysis-Based Federated Learning (CA-FL), presented in [9]. In the CA-FL framework, the server only communicates with the representative who achieved a higher level of accuracy in each cluster.We implemented a regression model in machine learning and evaluated and compared the CA-FL model using only the federated average (FedAvg) [6] for human activity recognition (HAR) datasets. In the current work, we have enhanced the original CA-FL framework with a dynamic clustering scheme that reduces communication costs and more quickly ensures global model convergence. The result of the improvements is a new version of a deep learning-based framework called FedCO. In contrast to the original framework and compared to related work studies, discussed in Section 2, FedCO incorporates the following amendments.

- We propose a deep learning-based FL framework, FedCO for short, that employs a dynamic adaptation procedure to new data, which evaluates representatives tied to their clusters at each learning round and redistributes them among the clusters if necessary. In addition, the quality of the obtained adapted clustering is evaluated at each round, and over-represented clusters of workers undergo a splitting procedure if this improves the whole clustering (Section 4).
- We provide a convergence analysis for our proposed FedCO algorithm (Section 6.2).
- We initially evaluate the proposed FedCO by comparing its performance with that of three baseline FL methods—FedAvg [6], FedProx [10], and CMFL [11]—on MNIST, CIFAR-10, and Fashion-MNIST under two different data-distribution scenarios, Independent and Identically Distributed (IID), and Non-IID.
- In addition, since our proposed FedCO algorithm is intended as a communication-mitigated version of FedAvg, we further study and assess the robustness of the FedCO with respect to FedAvg on two LEAF datasets under IID and Non-IID data.
- The conducted experiments have demonstrated the efficiency of FedCO over the FedAvg, FedProx, and CMFL algorithms in terms of convergence rate and communication overhead (Section 6).

The rest of the paper is structured as follows. Section 2 reviews the previous studies related to our work. The methodology used in our paper is presented in Section 3. Section 4 is devoted to the proposed FedCO and its strategy. The practical applications of those experimental settings are discussed in Section 5. The conducted experiments and the obtained results are analyzed and discussed in Section 6. The conclusions of our study and potential future works are presented in Section 7.

2. Related Work

This section mainly reviews the published research works aimed at reducing communication overheads in FL. In general, Federated Learning requires massive communication between the central server and the workers to train a global model [6]. Such an overhead is imputed to the size of the model exchanged and to the number of rounds to converge. Many works aim at reducing communication costs; e.g., HeteroFL [12] utilizes models of different sizes to address heterogeneous clients equipped with different computation and communication capabilities, while the work in [13] uses decentralized collaborative learning in combination with the master–slave model.

Among many of the published FL solutions, there are few existing FL works that use clustering techniques [14–18]. For example, in [14] the study proposes clustering algorithms based on clients' similarities. The authors have tried to find a cluster structure of data to collect clients with similar data distributions and to perform baseline FedAvg training per cluster. In [15], the authors introduce clustering techniques to partition the clients with similar data distribution using a measure of distance between the weight updates of the clients. A dynamic clustering through generative adversarial network-based clustering (GAN) is designed to obtain a partition of the data distributed on FL clients in [16]. The authors in [17] introduced a new framework, namely the Iterative Federated Clustering Algorithm (IFCA), in which clusters of users also aggregate their data with others in the same cluster (the same learning task) and optimize model parameters for the user clusters via gradient descent. Finally, Ouyang et al. [18] present clustering algorithms to cluster the heterogeneous data across clients into various clusters to participate in global model learning. The authors grouped the data after reducing its dimensions using PCA, and they measured the similarity of local updates.

Although the studies discussed above [14–18] have applied clustering techniques to FL scenarios, all of them have clustered the clients based on the distribution of their own data, while our proposed technique partitions the clients based on their training model parameters, i.e., in a way that ensures that each cluster will contribute to the model by learning different aspects (different model parameters' values) of the studied phenomenon. Evidently, our solution for mitigating communication costs of FL is conceptually different from the approaches discussed above, despite it also being based on clustering.

The majority of the studies in the field of resource-aware FL can be distributed into two main categories: a reduction in the total number of bits transferred, and a reduction in the number of local updates. Table 1 summarizes the techniques proposed by the research community, classifying them according to the categorization mentioned above.

Table 1. Summary of recent studies to minimize communication overhead in FL.

Categories	Existing Studies	ML Model Used	Datasets
First category	Chen et al. [19]	CNN, LSTM	MNIST, HAR
	Fed-Dropout [20]	DNN	CIFAR-10, MNIST, EMNIST
	Lin et al. [21]	CNNs, RNNs	Cifar10, ImageNet, Penn Treebank
	STC [22]	VGG11, CNN	CIFAR-10, MNIST
	PowerSGD [23]	ResNet-18, LSTM	CIFAR10, WIKITEXT-2
	FedOpt [24]	NN, LM	CIFAR10, MNIST
	FEDZIP [25]	CNN, VGG16	MNIST, EMNIST
	FetchSGD [26]	NN	CIFAR-100, CIFAR-10, FEMNIST
	T-FedAvg [27]	MLP, ResNet-18	MNIST, CIFAR-10
	FedAT [28]	CNN, Logistic	CIFAR-10, Fashion-MNIST, Sentiment140, FEMNIST, Reddit

Table 1. *Cont.*

Categories	Existing Studies	ML Model Used	Datasets
Second category	CMFL [11]	CNN, LSTM	MNIST, NWP
	FedMed [29]	LSTM	PTB, WikiText-2, Yelp
	CEEP-FL [30]	CNN	MNIST, CIFAR-10
	FedCS [31]	NN	CIFAR-10, FashionMNIST
	FedPSO [32]	CNN	MNIST, CIFAR-10
	AdaFL [33]	MLP, CNN	MNIST, CIFAR-10
	MAB [34]	NN, CNN	MNIST, Video QoE
	FedAtt [35]	GRU	WikiText-2, PTB, Reddit
	FedPAQ [13]	CNN, Logistic	MNIST, CIFAR-10
	Ribero et al. [36]	CNN, Logistic, RNN	Synthetic, EMNIST, Shakespeare
	CA-FL [9]	SGD	mHealth, Pamap2
	Proposed (FedCO)	CNN	MNIST, Fashion-MNIST, CIFAR-10, FEMNIST, CelebA

2.1. Reduction of the Total Number of Bits

The first category incorporates works that reduce the total number of bits transferred for each local update through data compression. Chen et al. [19] propose an enhanced Federated Learning technique by introducing an asynchronous learning strategy on the clients and a temporally weighted aggregation of the local models on the server. Different layers of the deep neural networks are categorized into shallow and deep layers, and the parameters of the deep layers are updated less frequently than those of the shallow layers. In addition, a temporally weighted aggregation strategy is applied on the server to make use of the previously trained local models, thereby enhancing the accuracy and convergence of the central model. Caldas et al. [20] design two novel strategies to reduce communication costs. The first relies on lossy compression on the global model sent from the server to the client. The second strategy uses Federated Dropout, which allows users to efficiently train locally on smaller subsets of the global model and reduces client-to-server communication and local computation. Lin et al. [21] propose Deep Gradient Compression (DGC) to significantly reduce the communication bandwidth. Sattler et al. [22] introduce a new compression framework, entitled Sparse Ternary Compression, that is specifically designed to meet the requirements of the Federated Learning environment. Asad et al. [24] implement a Federated Optimization (FedOpt) approach by designing a novel compression algorithm, entitled Sparse Compression Algorithm (SCA), for efficient communication, and then they integrate the additively homomorphic encryption with differential privacy to prevent data from being leaked. Malekijoo et al. [25] develop a novel framework that significantly decreases the size of updates while transferring weights from the deep learning model between the clients and their servers. A novel algorithm, namely FetchSGD, that compresses model updates using a Count Sketch and takes advantage of the mergeability of sketches to combine model updates from many workers, is proposed in [26]. Xu et al. [27] present a federated trained ternary quantization (FTTQ) algorithm, which optimizes the quantized networks on the clients through a self-learning quantization factor. Vogel et al. [23] design a PowerSGD algorithm that computes a low-rank approximation of the gradient using a generalized power iteration. A novel Federated Learning method, entitled FedAT, with asynchronous tiers under Non-IID data, is presented in [28]. FedAT synergistically combines synchronous intra-tier training and asynchronous cross-tier training. By bridging the synchronous and asynchronous training through tiering, FedAT minimizes the straggler effect with improved convergence speed and test accuracy. Our research does not consider methods that leverage data compression techniques because of

reduced scalability in scenarios such as edge and fog computing, and 5G networks, where hundreds of thousands of nodes cooperate in updating global models on the central server. Moreover, these approaches strictly depend on the application field.

2.2. Reduction of the Number of Local Updates

The second category includes studies that aim at reducing the number of local updates during the training process. For example, Wu et al. [29] have proposed a novel FedMed method with adaptive aggregation using the topK strategy to select the top workers who have lower losses to update the model parameters in each round. Likewise, Asad et al. [30] have provided a novel filtering procedure on each local update that allows transferring only the significant gradients to the server. The authors in [11] identify the relevant updates of the participants and upload only them to the server. In particular, at each round, the participants receive the global tendency and check the relevancy of their local updates with the global model, and only upload them if they align. Nishio and Yonetani in [31] propose an FL protocol of two-step client selection based on their resource constraints instead of the random client selection. In addition, a global model update algorithm, namely FedPSO, proposed transmitting the model weights only for the client that has provided the best score (such as accuracy or loss) to the cloud server [32].

Notice that our proposed FL model falls into the second category. We have been inspired by the studies discussed above, especially by CMFL [11] and FedProx [10], and we explored an approach that applies clustering optimization to bring efficiency and robustness in FL's communication. The most representative updates are uploaded only to the central server to reduce network communication costs.

The state-of-the-art solutions analyzed mainly conduct experiments considering a CNN model, except for FedMed, which uses an LSTM model, and FedCS, which uses an NN model (cf. Table 1, second category). Hence, we have chosen to assess the performance of our approach (FedCO) by using a CNN model. While there are many datasets used for the evaluation of FL solutions in the literature, the recurrent ones are MNIST, FashionMNIST, and CIFAR-10. Hence, we have evaluated the performance of FedCO training the FL model on the three datasets mentioned above. Additionally, we used datasets from the LEAF FL repository (FEMNIST and CelebA) to benchmark the performance of our FL algorithm against FedAvg [6] and FedProx [10].

3. Preliminaries and Definitions

In this section, we first briefly present the communication model and describe some preliminaries of a naive method of FL [37]. We then describe three state-of-the-art FL algorithms used for the comparison of our solution. Finally, we introduce the techniques used to conduct clustering optimization, i.e., the *k*-medoids clustering algorithm, and the Silhouette Index validation method. Table 2 summarizes the main notations used in the paper.

Table 2. Main notations.

Notation	Description
W	Set of available workers
W_t	Set of selected workers at tth communication round
w_i	A worker, i.e., $w_i \in W$
$\mathcal{D}_i$	The local data in worker w_i
n_i	The size of data in worker w_i
n	Total size of data
k_t	The number of clusters in round t

Table 2. *Cont.*

Notation	Description
$C = \{C_1, \ldots, C_{k_t}\}$	The clustering solution in round t
$\mathcal{M}$	The global model
$\mathcal{M}^*$	The optimal global model
$\mathcal{M}_t$	The global model at tth round
$\mathcal{M}_t^i$	The local model of worker w_i at round t
$F(.)$	The objective function of the global model
$F_i(.)$	The objective function of the local model of worker w_i
T	Maximal number of communication rounds
E	The number of local epochs
η	Learning rate
g_t^i	The gradients computed using back-propagation
$s(.)$	Silhouette Index score

3.1. Communication Model

In the proposed FL environment, FL is split into two major parts: workers and the central server. Our work aims to reduce communication overhead without sacrificing accuracy value during the training process. In this setting, the server coordinates a network of workers, controls the training progress of the model, broadcasts the original model to all participating workers, and then executes all the aggregation processes of the model updates. All workers are share model updates instead of sending their private data to a central server for global model aggregation. Figure 1 outlines the overall operations of the Federated Learning procedure.

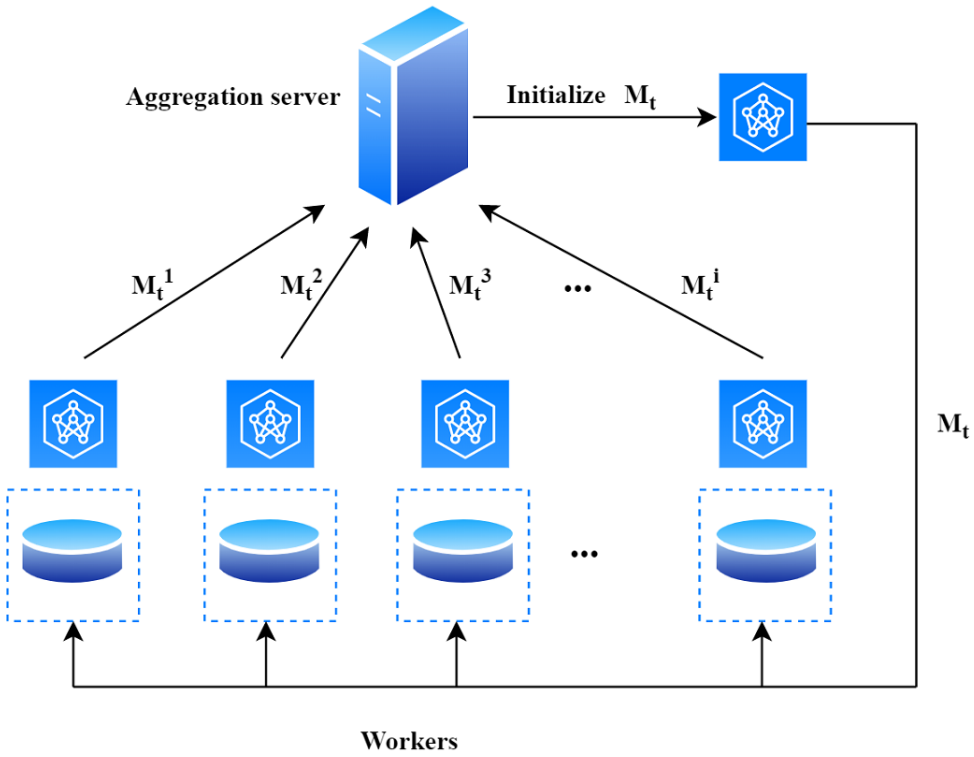


Figure 1. The general operations of the Federated Learning process.

Data are protected, with private access for each worker. Thus, model training occurs locally on each worker's side. In this context, we assume that each worker agrees on the same learning task and the model parameters throughout the training process. In particular, the proposed FL model updates the global model only with local model parameters from a few workers that are considered representative. Such workers are selected at each training round by identifying the highest quality of the local model produced of the worker. The selection policy is assumed to be implemented in a server, i.e., a central node selects a representative of the cluster with the highest accuracy. Furthermore, we assume that the server is always reachable by the workers. Finally, our proposed technique works by following this iterative collaboration between the central server and the workers.

3.2. Problem Description

In this work, we mainly concentrate on synchronous Federated Learning algorithms. A Federated Learning system consists of a global model $\mathcal{M}$ and a set of workers W. At each communication round t, the server deploys the current model $\mathcal{M}_t$ to a subset of workers $W_t \subset W$ that dynamically participate in the global aggregation at round t. Each worker $w_i \in W_t$ locally keeps its personal data $\mathcal{D}_i = \{x_{ij}\}_{j=1}^{n_i}$, $(j = 1, 2, \ldots, n_i)$, where x_{ij} is the jth training sample in $\mathcal{D}_i$. The size of the local dataset $\mathcal{D}_i$ varies with different real-world applications.

In standard centralized Stochastic Gradient Descent (SGD), the local updates of each w_i are calculated according to Equation (1) to optimize $\mathcal{M}_t^i$, where η is the learning rate and g_t^i refers to the gradients computed:

$$\mathcal{M}_{t+1}^i = \mathcal{M}_t^i - \eta g_t^i. \tag{1}$$

Then, each worker w_i sends the local model changes $\mathcal{M}_{t+1}^i$ to the central server after the number of E local step, where p_i is the relative weight of worker w_i, and the global model is computed by applying Equation (2):

$$\mathcal{M}_{t+1} = \mathcal{M}_t + \frac{\sum\limits_{w_i \in W_t} p_i \mathcal{M}_{t+1}^i}{\sum\limits_{w_i \in W_t} p_i}. \tag{2}$$

These are iterated until a certain stop criterion is met.

The corresponding local loss function of $\mathcal{M}^i$ of each worker w_i is defined as

$$F_i(\mathcal{M}^i) = \frac{1}{|\mathcal{D}_i|} \sum_{x_{ij} \in \mathcal{D}_i} f(\mathcal{M}^i, x_{ij}), \tag{3}$$

where $f(\mathcal{M}^i, x_{ij})$ is the loss function for data point x_{ij} using (1). Each worker w_i independently updates the model over its own data $\mathcal{D}_i$ to optimize its local loss function $F_i(\mathcal{M}^i)$. The aim of improving the communication efficiency of Federated Learning is to minimize the cost of sending $\mathcal{M}_t^i$ to a central server while learning from the data distributed over a large number of decentralized edge devices. Similarly, the global loss function on all the distributed datasets is defined as:

$$F(\mathcal{M}) = \frac{1}{\mid W \mid} \sum_{w_i \in W_t} F_i(\mathcal{M}^i), \tag{4}$$

where $\mathcal{M}$ is the aggregated global model, and the overall goal is to decrease the global loss function $F(\mathcal{M})$, namely,

$$\mathcal{M}^* = \arg\min F(\mathcal{M}). \tag{5}$$

Other issues related to Federated Learning problems, such as system heterogeneity or privacy, are beyond the scope of this paper. Specifically, the proposed FedCO algorithm does not account for heterogeneity, which for example could affect the selection of workers that have enough power to transmit the model parameters. In the worst case, heterogeneity could increase the convergence time or reduce the accuracy, if for example, workers that achieve a higher accuracy cannot be selected because they have short battery lifetimes.

3.3. FL State-of-the-Art Algorithms

Most of the work on the convergence of compared FL algorithms such as FedAvg, CMFL, and FedProx centers around minimizing (4). We compare the proposed FedCO with the following state-of-the-art algorithms in the FL setting:

3.3.1. *FedAvg*

FedAvg, proposed by McMahan et al. in [6] can be viewed as a communication-light implementation of the standard centralized SGD, wherein the local updates are aggregated in the server after E local steps, where $E \geq 1$.

3.3.2. *FedProx*

FedProx [10] is a distributed algorithm, wherein a round-varying proximal term is introduced to control the deviation of the local updates from the most recent global model. A participating worker uses a proximal update that involves solving a minimization problem.

3.3.3. *CMFL*

Communication-Mitigated Federated Learning (CMFL) [11] improves the communication efficiency of Federated Learning while at the same time providing guaranteed learning convergence.

3.4. K-Medoids Clustering Algorithm

K-medoids is a robust clustering algorithm. It is used to partition a given set of data points into k disjoint clusters [38]. In contrast to the k-means, which use the mean value of the data points in each cluster as a cluster centroid, k-medoids chooses an actual data point, called a medoid. The medoid is the most centrally located point in a given cluster. Therefore, k-medoids are more robust to outliers and noise than other points. The algorithm works by arbitrarily choosing a set of k initial cluster medoids from a given set of data points, where k is preliminarily specified. Then, each data point is assigned to the cluster whose center is the nearest, and the cluster centers (medoids) are recomputed. This process is repeated until the points inside every cluster become as close to the center as possible, and no further item reassignments take place.

In our FedCO algorithm, we use k-medoids for partitioning the available workers into groups of similar workers with respect to their local updates. Furthermore, 2-medoids are used in the iteration phase of the algorithm for conducting cluster splitting.

3.5. Silhouette Index

The *Silhouette Index* (SI) is a widely used internal cluster validation technique, introduced in [39]. SI can be used to judge the quality of any clustering solution $C = \{C_1, C_2, \ldots, C_k\}$. It assesses the separation and compactness between the clusters. Suppose that a_i represents the average distance of item i from all the other items in the cluster to which item i is assigned, and b_i represents the minimum of the average distances of item i from the items of the other clusters. Then, the *Silhouette score* $s(i)$ of item i can be calculated as

$$s(i) = (b_i - a_i)/\max\{a_i, b_i\}. \quad (6)$$

$s(i)$ measures how well item i matches the clustering at hand. $s(i) \in [-1, 1]$, and if $s(i)$ is close to 1, this means that item i is assigned to a very appropriate cluster. The situation is

different when $s(i)$ is near zero. Specifically, item i lies between two clusters. The worst case is when $s(i)$ is close to -1. Evidently, this item has been misclassified.

In addition, the overall Silhouette score for the whole clustering solution C of n items is determined as

$$s(C) = \frac{1}{n} \sum_{i=1}^{n} \frac{(b_i - a_i)}{\max\{a_i, b_i\}}. \quad (7)$$

The SI can also be calculated for each cluster C_j ($j = 1, 2, \ldots, k$) of n_j objects as follows:

$$s(C_j) = \frac{1}{n_j} \sum_{i=1}^{n_j} s(i). \quad (8)$$

The FedCO algorithm proposed in this study uses the Silhouette Index at each iteration round for assessing the current workers' partitioning and, based on this assessment, selects what optimizing actions to conduct. For example, we used SI to check whether a representative is still firmly tied to its current cluster of workers. It may happen that some representatives will change their clusters. If we have a worker that produces a negative SI value for all clusters, this means that this worker cannot be assigned to any of the existing clusters, and it will form a new singleton cluster; i.e., a new concept appears. In addition, SI is applied to assess whether an intended splitting of a cluster will improve the quality of the whole clustering solution, i.e., whether it should be conducted. For more details, see Section 4. Note also that in the implemented version of our FedCO algorithm, we use Euclidean distance to measure the similarity between each pair of workers. In particular, the Euclidean distance between the worker (the representative) and the cluster centers (medoids) has been computed.

4. Proposed Approach

Our proposed FedCO algorithm foresees two distinctive phases: *initialization* and *iteration*. These phases are described in what follows, along with cluster optimization algorithms. In addition, the algorithm pseudo-code is reported in Algorithms 1 and 2.

Let $W = \{w_1, w_2, \ldots, w_n\}$ be the set of all available workers, and W_t is a subset of W that contains the workers selected at round t. The workers in W_t can be the representatives of the clusters $C_t = \{C_{t1}, C_{t2}, \ldots, C_{tk_t}\}$ obtained by applying a clustering algorithm to W, or a set of randomly selected workers, and $\mid W_t \mid < n$.

Algorithm 1 Federated Learning Using Clustering Optimization (FedCO)

Output: The FEDCO procedure updates the global model $\mathcal{M}_t$ for T iterations

procedure FEDCO($\mathcal{M}_0$, $W_t \subseteq W$, k_t, T)
Initialization Phase
  $t \leftarrow 0$
  $\forall\, w_i \in W_t$, SEND(w_i, $\mathcal{M}_t$)
  for each worker $w_i \in W_t$ *in parallel* **do**
    $\mathcal{M}^i_{t+1} \leftarrow$ WORKERUPDATE(i, $\mathcal{M}_t$)
  end for
  $\mathcal{M}_{t+1} = \sum_{w_i \in W_t} \frac{n_i}{n} \mathcal{M}^i_{t+1}$ following (2)
  $C_t \leftarrow$ KMEDOIDS(k_t, $\{\mathcal{M}^i_{t+1} \mid w_i \in W_t\}$, W_t)
Iteration Phase
  while $t \leq T$ **do**
    $t \leftarrow t + 1$
    $W_t \leftarrow$ SELECTTOPRANKED(p, C_t)
    $\forall\, w_i \in W_t$, SEND(w_i, $\mathcal{M}_t$)
    for each worker $w_i \in W_t$ *in parallel* **do**
      $\mathcal{M}^i_{t+1} \leftarrow$ WORKERUPDATE(w_i, $\mathcal{M}_t$)
    end for
    $\mathcal{M}_{t+1} = \sum_{w_i \in W_t} \frac{n_i}{n} \mathcal{M}^i_{t+1}$
    $C_{t+1} \leftarrow$ SILHOUETTE(k_t, C_t, W_t)
    while $\mid C_{t+1} \mid < \mid C_t \mid$ **do**
      $C_{t+1} \leftarrow$ CLUSTERINGOPTIMIZATION(k_{t+1}, C_{t+1})
    end while
  end while
end procedure

function SILHOUETTE((k_t, C_t, W_t)) ▷ *Check whether each cluster representative still belongs to its cluster*
  for $w_i \in W_t$ **do**
    for $j = 1, 2, \ldots, k$ **do**
      compute $s(w_i)$ ▷ *According to Equation (6)*
    end for
    if $s(w_i) < 0, \forall j \in \{1, 2, \ldots, k\}$ **then**
      $k_t \leftarrow k_t + 1$
      $C_{tk_t} \leftarrow w_i$
    else
      Assign w_i to the nearest cluster C_{tj}
    end if
  end for
  $\forall\, C_{tj}$ $(j = 1, 2, \ldots, k)$ recompute the cluster center
  return C_{t+1} ▷ *The new set of clusters*
end function

function WORKERUPDATE((w_i, $\mathcal{M}_t$)) ▷ *Local update*
  while True **do**
    RECEIVE(w_i, $\mathcal{M}_t$)
    LOCALTRAINING(w_i, $\mathcal{M}_t$)
    $\mathcal{M}^i_{t+1} \leftarrow \mathcal{M}^i_t - \eta g^i_t$ ▷ *Local update, (1)*
    SEND(i, $\mathcal{M}^i_{t+1}$)
  end while
end function

Algorithm 2 ClusteringOptimization

Output: updated k_t and C_t

procedure ClusteringOptimization(k_t, C_t)
  $s(C_t) \leftarrow$ SilhouetteScore(k_t, C_t, W_t)
  $C'_t \leftarrow \emptyset$
  for $C_{tj} \in C_t$ *s.t.* $\mid C_{tj} \mid > 1$ **do**
    $s(C_{tj}) \leftarrow$ SilhouetteCluster(C_{tj}, W_t)
    while $s(C_{tj}) < 0$ **do**
      $(C^1_{tj}, C^2_{tj}) \leftarrow$ Kmedoids($\{\mathcal{M}^i_{t+1} \mid w_i \in C_{tj}\}, k = 2$) ▷ *run 2-medoids to generate two new clusters*
      $\bar{C}_t \leftarrow \{C_t \setminus \{C_{tj}\}\} \cup \{C^1_{tj}, C^2_{tj}\}$
      $s(\bar{C}_t) \leftarrow$ SilhouetteScore(k_t, C_t, W_t)
      if $s(\bar{C}_t) > s(C_t)$ **then**
        $C'_t \leftarrow C'_t \cup \bar{C}_t$
        $k_t \leftarrow k_t + 1$
      end if
    end while
  end for
  return (C'_t, k_t)
end procedure

function SilhouetteScore(k_t, C_t, W_t) ▷ *Silhouette score of whole cluster solution* C_t
  Compute $s(w_i)$ between each $w_i \in W_t$ and each medoid $c_{tj} \in C_{tj}$ $(j = 1, 2, \ldots, k_t)$ according to (6)
  Compute the average Silhouette score over all representatives $w_i \in W_t$ according to (7)
  return $s(C_t)$
end function

function SilhouetteCluster(C_{tj}, W_t) ▷ *Silhouette Score of cluster* $C_{tj} \in C_t$ $(j = 1, 2, \ldots, k)$
  Calculate Silhouette score $s(w_i)$ for each $w_i \in C_{tj}$ according to (6)
  Compute the mean over Silhouette scores of all cluster members $\{s(w_i) \mid w_i \in C_{tj}\}$ according to (8)
  return $s(C_{tj})$
end function

4.1. Initialization Phase

1. At time $t = 0$, the Server initializes the inputs for the FedCO algorithm (Algorithm 1). These are the model $\mathcal{M}_0$, the set of representative workers W_t, the number of clusters k_t, and the number of iterations T. $t = 0$ (line 1 in Algorithm 1).
2. A central Server transmits the initial global model $\mathcal{M}_t$ to a set of workers W_t ($W_t \subset W$). These are selected to be used for initial training in round $t = 0$ of Federated Learning (lines 3 in Algorithm 1).
3. Each worker $w_i \in W_t$ receives the global model $\mathcal{M}_t$ and optimizes its parameters locally; i.e., the $\mathcal{M}^i_t$ initial update is produced and sent back to the Server (Equation (1)) (lines 4–6 and lines 38–45 in Algorithm 1).
4. The Server aggregates the parameters $\{\mathcal{M}^i_t \mid w_i \in W_t\}$ uploaded by the selected workers W_t to update the global model $\mathcal{M}_t$ through the FedAvg algorithm (Equation (2)) (line 7 in Algorithm 1).
5. The local updates $\{\mathcal{M}^i_t \mid w_i \in W_t\}$ of the workers in W_t are analyzed by using the k-medoids clustering algorithm (function KMedoids, line 8 in Algorithm 1)). As a result, k_t clusters of workers with similar updates are obtained; i.e., an initial clustering $C_t = \{C_{t1}, C_{t2}, \ldots, C_{tk_t}\}$ of the workers in W_t is produced.

4.2. Iteration Phase

1. At each iteration round t ($t \geq 0$), the Server evaluates each local update $\mathcal{M}_t^i$, $w_i \in W_t$ by using an evaluation measure that is suitable for the task under consideration. It ranks the workers in each cluster $C_{tj}, j = 1, 2, \ldots, k_t$ with respect to their evaluation scores and selects the top-ranked worker, i.e., the representative (function SELECTTOPRANKED, line 11 in Algorithm 1). The selected representatives form a new set of workers W_{t+1}, where $\mid W_{t+1} \mid = k_t$ and $k_t << \mid W_0 \mid$. Each selected worker $w_i \in W_{t+1}$ will check in with the Server.
2. The Server sends the global model $\mathcal{M}_t$ to each representative $w_i \in W_{t+1}$ (line 12 in Algorithm 1).
3. Each representative $w_i \in W_{t+1}$ receives the global model $\mathcal{M}_t$ and optimizes its parameters locally; i.e., the $\mathcal{M}_{t+1}^i$ update is produced (Equation 1) and sent back to the Server (lines 13 and 15 in Algorithm 1).
4. The Server aggregates the received local models $\{\mathcal{M}_{t+1}^i \mid w_i \in W_{t+1}\}$ uploaded by the representatives to update the global model through the FedAvg algorithm; i.e., an updated global model $\mathcal{M}_{t+1}$ is produced (Equation (2)) (line 16 in Algorithm 1).
5. The Server adapts C_t to the newly arrived local updates by conducting the following operations:
   (a) SI invokes the SILHOUTTE function (lines 17, 23–37 in Algorithm 1), which assesses whether each representative $w_i \in W_{t+1}$ is still adequately tight with its current cluster (Equation (6)). The updated clustering C_{t+1} of W_t is produced, and the clusters in C_{t+1} may contains a set of workers different from C_t. Note that $k_{(t+1)} \geq k_t$, where $k_{(t+1)} = \mid C_{t+1} \mid$, since new singleton clusters may appear due to the updating operation. This happens when the Silhouette coefficient $s(w_i)$ of a representative for all clusters gives a negative value (lines 28–30 in Algorithm 1), which means that this representative cannot be assigned to any existing cluster. Hence, this representative could be considered as a new cluster with a single item (singleton).
   (b) If there is a cluster $C_{(t+1)j} \in C_{t+1}$, such that $C_{(t+1)j} = \emptyset$, then $C_{t+1} = C_{t+1} \setminus \{C_{(t+1)j}\}$, and therefore, $\mid C_{t+1} \mid < \mid C_t \mid$. This condition/event triggers the optimization of the number of clusters by invoking the CLUSTEROPTIMIZATION function (lines 18–20 in Algorithm 1). This operation is repeated for each empty cluster of C_{t+1}.

A schematic illustration (flowchart) of the overall processes of the proposed FedCO algorithm is given in Figure 2.

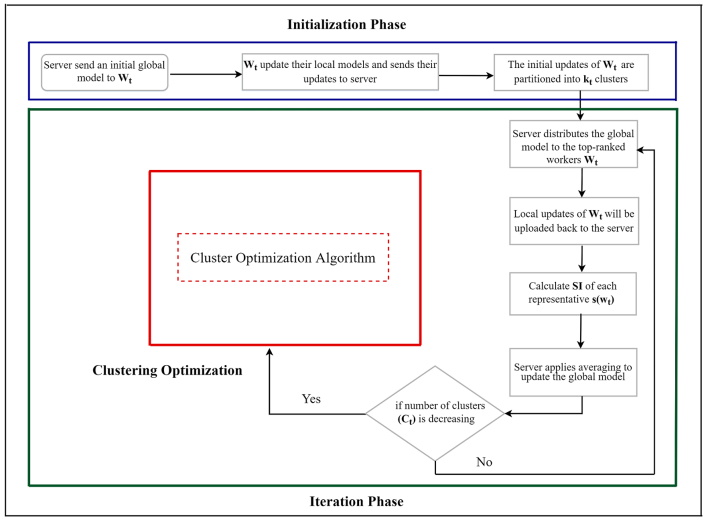


Figure 2. A schematic illustration of the entire process of the FedCO algorithm in two global communication rounds: Initialization Phase and Iteration Phase.

4.3. Cluster Optimization

The CLUSTEROPTIMIZATION algorithm works in what follows (cf. Algorithm 2):

1. Firstly, the SI score of the whole clustering solution C_{t+1} is computed. This score is used to check whether the splitting operation really improves the quality of the clustering solution (line 2 in Algorithm 2).
2. Then, the SI score is calculated for each cluster $C_{(t+1)j} \in C_{t+1}$, such that $\mid C_{(t+1)j} \mid > 1$ using Equation (8). If $s(C_{(t+1)j}) < 0$, then this cluster is a candidate to be split into two clusters, and the following operations are performed (lines 4–6 in Algorithm 2):
   (a) The two most distant points in the cluster $C_{(t+1)j}$ are found. They are used to seed 2-medoids clustering, which is applied to split the cluster $C_{(t+1)j}$ into two clusters (function KMEDOIDS at line 7 in Algorithm 2).
   (b) The clustering solution C_{t+1} is updated by replacing cluster $C_{(t+1)j}$ with the two clusters obtained due to the splitting operation (line 8 in Algorithm 2), and stored in the set $\bar{C}_{t+1}$.
   (c) The SI score of the updated clustering solution $\bar{C}_{t+1}$ is computed (line 9 in Algorithm 2 (7)).
   (d) If the SI score of the new clustering solution $\bar{C}_{t+1}$ is higher than the one before splitting, the new clustering solution is adopted and stored in the set C'_{t+1}; otherwise, the clustering solution C_{t+1} is kept (lines 10–13 in Algorithm 2).

Steps 1–5 of the *iteration* phase are repeated until a certain number of training rounds T is reached. Figure 3 shows a flowchart depicting the cluster optimization algorithm in a single round of communication.

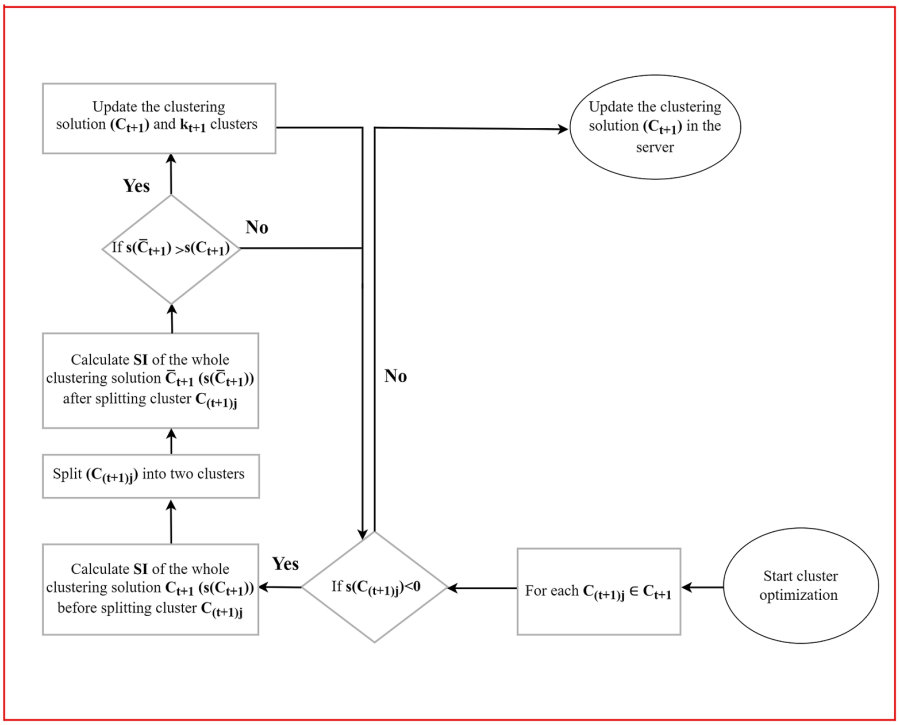


Figure 3. Flowchart depicting Cluster optimization algorithm.

The proposed FedCO implementation, at each training round, always selects the top performing representative; i.e., the size of the clusters is not reflected in the aggregated global model, and the size of the cluster does not impact the selection/importance of the representative. The FedCO design, however, allows from each cluster the selection of several top-ranked representatives, i.e., more than one, proportionally to the cluster size. In that way, the bigger clusters will have more weight in the building of the global model. It is also possible to assign explicit weights to the clusters representing their relative

importance, and calculated based on their size. Our future plans include the investigation of an optimized version of the FedCO algorithm where the importance of clusters will be considered in the aggregated model.

5. Datasets and Experimental Setup

This section describes the datasets, the distribution of the data across the edge nodes, the model selected and related parameters, and the performance metrics used for evaluating FedCO.

5.1. Datasets

We conducted experiments using a wide range of datasets. Firstly, we selected three benchmark datasets widely used for image classification: MNIST [40], Fashion MNIST [41], and CIFAR-10 [42].

- The MNIST dataset contains a 60,000-point training set and a 10,000 point test set with 10 classes. Each sample is based on a grayscale image of handwritten digits with a size of 28 $\times$ 28 pixels.
- The Fashion MNIST dataset comprises a 60,000-point training set and a 10,000 point testing set of images of fashion items with 10 different classes. Each image has dimensions of 28 $\times$ 28 in grayscale.
- The CIFAR-10 dataset consists of a 50,000-point training set and a 10,000-point testing set with images of objects from frogs to planes, where each image is 32 $\times$ 32 pixels in 10 classes.

Secondly, we considered two LEAF datasets [43], an open-source benchmark for Federated Learning.

- FEMNIST for 62-class image classification, which serves as a more complex version of the popular MNIST dataset [44].
- CelebA for determining whether the celebrity in the image is smiling, which is based on the Large-scale CelebFaces Attributes Dataset.

5.2. Data Distribution

In an FL context, the performance is affected by the distribution of the training data stored on the various workers. Interestingly, unlike other FL studies using clustering techniques, different degrees of non-IID data do not affect the clustering results, as FedCO clustering occurs based on the model parameters and not on the data themselves. In order to assess the impact of different data distribution scenarios, we generated two experimental datasets for each dataset introduced above:

- The IID dataset: Each worker holds the local data equal in size and label distribution.
- The Non-IID dataset: Each worker holds different data distributions in size and label distribution compared to the global dataset.

5.3. Model Selection and Parameters

We have compared the proposed FedCO algorithm against the FedAvg, CMFL, and FedProx algorithms using the Convolutional Neural Network (CNN) classifier as a training model. The CNN model we used consists of two 5 $\times$ 5 convolution layers with a ReLU activation and a final softmax output layer.

The baseline configuration parameters' values listed in Table 3 are shared among the four compared algorithms.

Table 3. Hyper-parameter configuration.

Hyper-Parameter	Value
Workers	100
Optimizer	SGD
Classes	10
Batch Size	50
Learning rate	0.15
Local epochs	10
Global rounds	200
Clusters	8
Non-IID degree	0.5

5.4. Performance Metrics

FL typically relies on a large number of edge devices, sometimes in the magnitude of millions, and due to the limited computing capabilities of those devices, decreasing the communication rounds or communication overhead is crucial during the training process. Hence, the performance metrics selected are the *Number of Communication Rounds*, the *Communication Overhead*, and the model *Accuracy*. The *Communication Overhead* is defined in [9] as

$$(N \times \mid W_s \mid) \times (2 \times T + 1),$$

where N is the size of the trained model in bytes, $\mid W_s \mid$ is the number of selected workers, and T is the total number of training rounds. We assume the size of the model updates to be fixed. However, other communication costs are negligible.

It is worth mentioning that the total communication overhead of FedCO can be calculated as the summation of the communication costs of the initialization stage and the iteration stage together.

6. FedCO Performance Evaluation and Analysis

In this section, we first study the clustering optimization scheme used for the dynamic adaptation of partitioning of workers' updates at each communication round. This adaptive behavior contributes to achieving robust communication in FL. The performance of the proposed FedCO is then evaluated and compared to three other existing FL approaches (FedAvg, FedProx, and CMFL) in terms of accuracy, communication rounds, and communication overhead.

Our proposed FedCO algorithm is a communication-optimized version of FedAvg. Therefore, we further evaluate these two algorithms by benchmarking them on two datasets from the LEAF Federated Learning repository, namely FEMNIST and CelebA. In addition, we further study our FedCO algorithm for two different scenarios for selecting cluster representatives: a performance threshold-based worker selection versus the single (top-performer) cluster representative selection, explained in Algorithm 1.

6.1. Clustering Optimization Behavior

Our clustering optimization algorithm assesses the local updates of clusters' representatives at each communication round, and as a result, it assigns some workers to different clusters. An output of this cluster-updating procedure is that clusters may appear or disappear. Our solution is capable of catching and handling these scenarios. In addition, it implements a splitting procedure that performs a further fine calibration of the clustering for the newly uploaded updates.

In order to illustrate the properties of the clustering optimization scheme discussed above, we show in Figure 4 the clustering updates in the first five global communication rounds of the FedCO algorithm applied to the Non-IID FashionMNIST dataset. In the example, in round 2, cluster 5 has disappeared and cluster 3 is a singleton, i.e., it cannot be a candidate for splitting. Almost all of the remaining clusters (except cluster 6) have negative SI scores. The remaining clusters (0, 1, 2, 4, and 7) have been split into two new clusters and

their cluster labels are replaced. Interestingly, in round 3, the unique number of clusters is 17. However, in round 4, five clusters have turned out empty and have disappeared (1, 4, 10, 13, and 16). Furthermore, eight clusters have positive SI scores (0, 2, 5, 6, 9, 12, 14, and 15), while four have negative SI scores (3, 7, 8, and 11). The algorithm did not split the clusters 7, 8, and 11 because this did not improve the quality of the clustering solution; i.e., it did not increase its SI score. Cluster 3 is still a singleton. The worker belonging to this cluster may be considered as one that provides unique model parameters due to its training data. Consequently, in round 5, we have only 12 clusters. Two clusters disappeared (2 and 14), and four new clusters appeared (1, 6, 12, and 17), while clusters 3 and 9 were singletons.

The cluster optimizations discussed above will continue in the same fashion for the upcoming communication rounds. The workers' partitioning is dynamically adapted at each communication round to reflect the new local updates of the representatives.

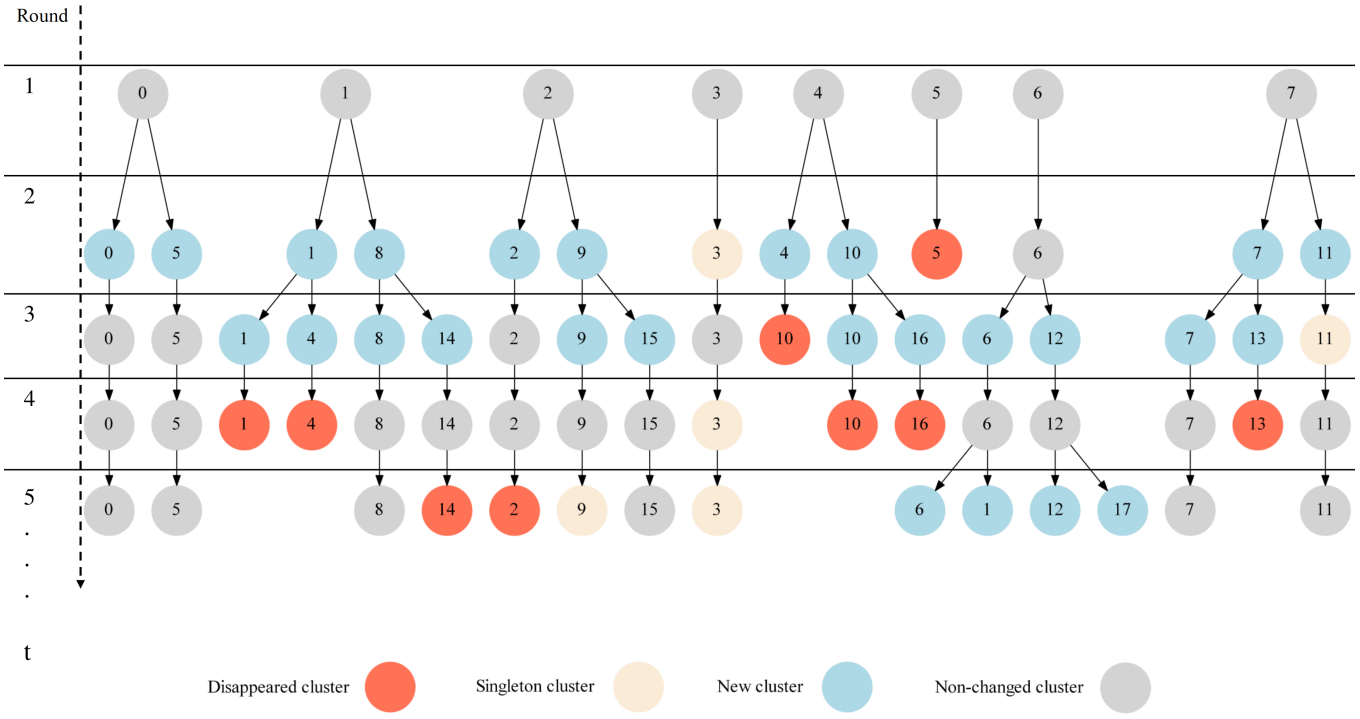


Figure 4. The clustering updates in the first five global communication rounds of the proposed FedCO algorithm applied on the Non-IID FashionMNIST dataset. Notice that the number in the circle represents the cluster label.

6.2. Convergence Analysis

In this section, we provide a convergence analysis of the proposed FedCO algorithm and theoretically show that it ensures a faster convergence than the baseline FedAvg algorithm.

Our analysis is based on two assumptions. The first one supposes that the data are non-IID. Secondly, we assume that there is a partial involvement of workers; this strategy is much more realistic as it does not require all of the worker output. Therefore, at each iteration, we can calculate the global update by aggregating the local updates by using those cluster representatives, which have reached a high accuracy level at this iteration phase. Two scenarios are considered to this end: (i) a global model is trained by FedAvg based on updates made by randomly selected workers, regardless of their accuracy value; (ii) a global model is trained by applying FedCO, and in that way, at each training round, only workers (cluster representatives) that have achieved the highest accuracy values are used.

Let us briefly summarize the working mechanism of the proposed FedCO algorithm. In the tth global training iteration, each worker involved ($w_i \in W_t$) calculates the average g_t^i gradients using the optimization algorithms in the local dataset in the current global model

$\mathcal{M}_t$. Note that according to Equation (5), high-quality data and a high accuracy of the workers' models can lead to a faster convergence of the local loss functions (Equation (3)) and the global loss function (Equation (4)) [45]. Both the local model update $\mathcal{M}_t^i$ of the worker in Equation (1) and the shared global model update $\mathcal{M}_{t+1}$ in Equation (5) can be more quick to converge to the target value with fewer iterations. Consequently, the training time of a worker in a global iteration is decreasing. Therefore, highly accurate workers' models can significantly improve the learning efficiency of Federated Learning; e.g., it can ensure less training time [31,46]. This process is iterative until a global accuracy ϵ ($0 \leq \epsilon \leq 1$) is achieved. Specifically, each update of the local model has a local accuracy w_i^ϵ that corresponds to the local quality of the worker w_i data. A higher local accuracy leads to fewer local and global iterations [46,47]. FedCO uses an iterative approach that requires a series of communication rounds to achieve a level of global accuracy ϵ. Server and representative communications occur during each global round of the iteration phase. Specifically, each representative minimizes its objective $F_i(\mathcal{M}^i)$ in Equation (3) using the local training data n_i. Minimizing $F(\mathcal{M})$ in Equation (4) also requires multiple local iterations up to a target accuracy. Then, the global rounds will be bounded as follows:

$$\frac{\mathcal{O}(\log(\frac{1}{\epsilon}))}{1 - w_i^\epsilon}$$

Thus, the global rounds are affected by both the global accuracy ϵ and the local accuracy w_i^ϵ. When ϵ and w_i^ϵ are high, FedCO needs to run a few global rounds. On the other hand, each global round consists of both computation and transmission time. Our primary motivation in this work is to consider the communication overhead, discussed and analyzed in detail in Section 6.3. The computation time (w_i^{cmp}), however, depends on the number of local iterations. When the global accuracy ϵ is fixed, the computation time is bound by $\log(\frac{1}{w_i^\epsilon})$ for an iterative algorithm to solve Equation 1; here, (SGD) is used [46]. Therefore, the total time of one global communication round for a set of representatives is denoted as

$$T^{com} = \sum_{w_i \in W_t} \log(\frac{1}{w_i^\epsilon}) w_i^{cmp} + w_i^{com},$$

where w_i^{com} represents the transmission time of a local model update. As a result, a high local accuracy value of w_i^ϵ leads to fewer local iterations w_i^{cmp} and eventually to lower global communication rounds T^{com}. Unlike FedCO's convergence rate, FedAvg does not necessarily guarantee a faster convergence speed. This is because FedAvg uses a much larger number of workers compared to the FedCO model. Therefore, if there are more workers with poor data quality, the convergence will be reached at a slower rate than when much fewer workers with high data quality are used. However, at each global round, FedCO may have selected a different set of workers. Those, however, are not selected randomly, but each one is a representative of a cluster of workers having modeled similar parameters, and in addition, it achieves the highest accuracy among the cluster members. Let T_{FedAvg} and T_{FedCO} represent the number of global rounds for which convergence has been reached by FedAvg and FedCO, respectively. Then, Tables 4 and 5 demonstrate that the inequality $T_{FedCO} < T_{FedAvg}$ is valid in the experiments aiming to reach the same accuracy using the two algorithms.

Table 4. The number of communication rounds to reach a target accuracy for the three compared FL algorithms.

	IID					
	MNIST		**FashionMNIST**		**CIFAR-10**	
	Rounds	**Saving**	**Rounds**	**Saving**	**Rounds**	**Saving**
FedAvg	190	(ref)	200	(ref)	200	(ref)
FedProx	185	26%	190	5%	188	6%
CMFL	50	73%	60	70%	80	60%
FedCO	25	86%	50	75%	30	85%
	Non-IID					
	MNIST		**FashionMNIST**		**CIFAR-10**	
	Rounds	**Saving**	**Rounds**	**Saving**	**Rounds**	**Saving**
FedAvg	170	(ref)	200	(ref)	200	(ref)
FedProx	167	17%	186	7%	>200	-
CMFL	60	64%	150	25%	160	20%
FedCO	30	82%	70	65%	60	70%

Table 5. The number of communication rounds to reach a certain accuracy level for the two compared FL algorithms on each LEAF dataset.

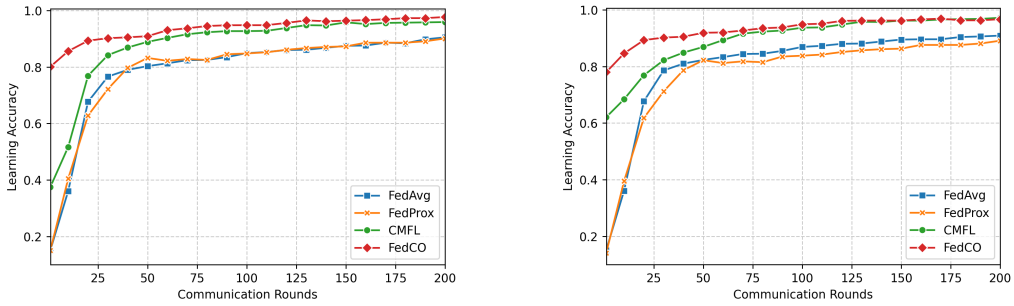
	IID			
	FEMNIST		**CelebA**	
	Rounds	**Saving**	**Rounds**	**Saving**
FedAvg	140	(ref)	110	(ref)
FedCO	12	91%	30	72%
	Non-IID			
	FEMNIST		**CelebA**	
	Rounds	**Saving**	**Rounds**	**Saving**
FedAvg	100	(ref)	150	(ref)
FedCO	14	86%	10	93%

6.3. Communication Rounds versus Accuracy

In this subsection, we present the results related to the evaluation of the accuracy of our distributed deep learning (DL) model. Figures 5–7 show how the compared FL (FedAvg, FedProx, CMFL, and FedCO) algorithms perform in terms of Accuracy versus the Number of Communication Rounds. For the MNIST dataset (see Figure 5), we can observe that the FedCO algorithm converges faster than with the state-of-the-art approaches. As is shown in Figure 5a (IID data distribution setting), FedAvg and FedProx use 100 rounds to obtain an accuracy of 85%. The CMFL reaches the same accuracy in 30 rounds, while our FedCO algorithm achieves this result with only 10 rounds. Furthermore, in Figure 5b, FedCO dramatically decreases the communication rounds with respect to FedAvg, FedProx, and CMFL. Indeed, in Non-IID data, a learning accuracy of 90% is achieved by FedCO in 40 rounds, FedAvg has conducted 160 rounds, FedProx requires 200 rounds, and CMFL needs 60.

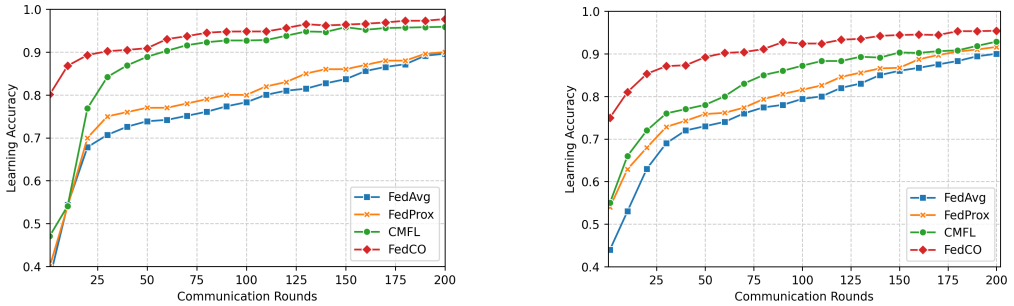
In Figure 6a, we compare the accuracy of the four FL approaches in the case of the IID data distribution scenario of FashionMNIST. The FedCO outperforms FedAvg, FedProx, and CMFL in this experimental setting. Within 25 communication rounds, CMFL, FedAvg, and FedProx reach 81%, 69%, and 74% accuracy, respectively, while our FedCO algorithm achieves an accuracy of 90% with the same number of communication rounds. Notice that under the Non-IID data distribution setting, our FedCO algorithm outperforms the other, reaching an accuracy of nearly 79% with only 11 rounds; this costs 100 communication

rounds for FedAvg and 80 rounds for FedProx. CMFL considerably minimizes this cost to 60; see Figure 6b.



(**a**) (**b**)

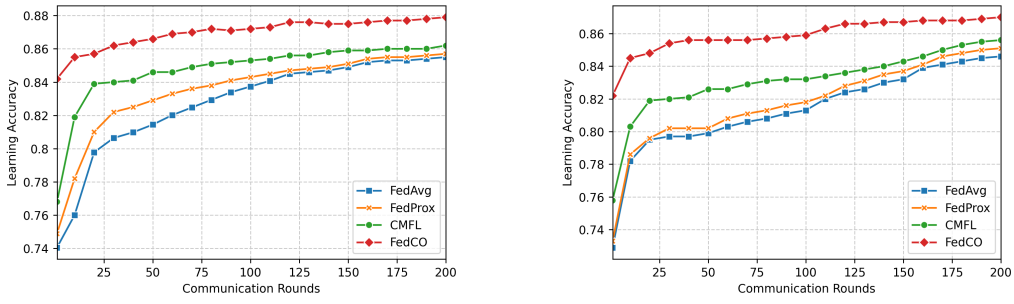
Figure 5. Learning accuracy versus the number of communication rounds for MNIST data. The top plot presents the results produced in the case of the IID data distribution scenario, while the bottom plot depicts the results generated in the case of the Non-IID data distribution scenario. (**a**) IID; (**b**) Non-IID.



(**a**) (**b**)

Figure 6. Learning accuracy versus communication rounds for FashionMNIST data. The top plot presents the results produced in the case of the IID data distribution scenario, while the bottom plot depicts the results generated in the case of the Non-IID data distribution scenario. (**a**) IID; (**b**) Non-IID.

Finally, for the CIFAR-10 IID data, the required communication costs of the FedAvg and FedProx to achieve 85% accuracy is 150 rounds, while CMFL obtains the same result for 75 rounds. Our FedCO algorithm outperforms the others, needing only nine rounds to reach this accuracy value (cf. Figure 7a). In the case of the CIFAR-10 Non-IID data (see Figure 7b), in 25 communication rounds, FedCO obtains an accuracy of 85%, while FedAvg and FedProx reach 79% and 80%, respectively. On the other hand, CMFL achieves a close result 82% of accuracy in the same number of rounds.



(**a**) (**b**)

Figure 7. Learning accuracy versus communication rounds for the CIFAR-10 data. The top plot presents the results produced in the case of the IID data distribution scenario, while the bottom plot depicts the results generated in case of the Non-IID data distribution scenario. (**a**) IID; (**b**) Non-IID.

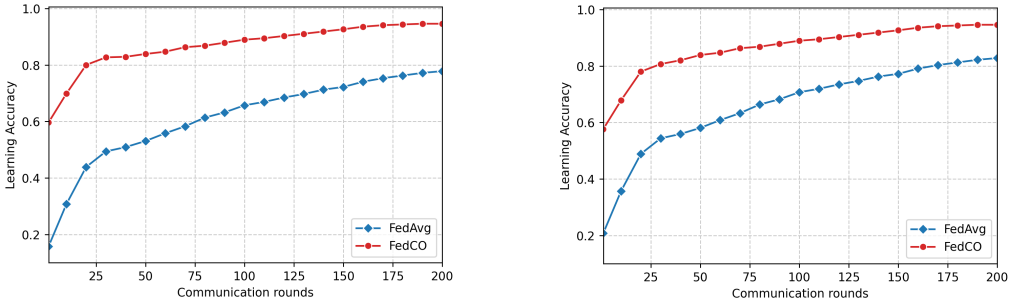
FedCO differs from the Federated Learning baseline FedAvg as follows: our algorithm uses a much smaller number of nodes while the aggregation procedure is the same. Thus, if we have a smaller number of workers, convergence is reached faster than in FedAvg, where all the available workers are used. At each round, FedCO selects and uses a different set of workers, and each worker is a representative that achieves the highest accuracy in each cluster. Hence, the accuracy is not sacrificed.

Table 4 shows the number of communication rounds to achieve the maximum model accuracy (i.e., to converge) for the datasets considered. Specifically, the target accuracy values are 90% for MNIST and FashionMNIST and 85% for CIFAR-10. FedAvg is the baseline benchmark, and the iterations saved for algorithm X (X = FedProx, CMFL, or FedCO) is computed as

$$1 - \frac{num\_of\_iteration\_X}{num\_of\_iteration\_FedAvg}.$$

FedCO saves from 75% to 86% of iterations to converge with respect to FedAvg for IID data distribution setting, and it saves from 65% to 82% iterations for the Non-IID data distribution scenario. Moreover, FedCO always converges with at least half of the iteration rounds needed by CMFL. In more detail, one can observe that the model on the MNIST IID data distribution setting converges to an accuracy of 90% in 190 rounds with the FedAvg algorithm, and in 25 rounds for our FedCO algorithm, providing savings of 86%, and in 185 rounds for FedProx, and in 50 rounds for CMFL, providing savings of 26% and 73%, respectively. The model trained on the FashionMNIST IID data distribution scenario converges to a target accuracy of 90% in 200 rounds for FedAvg, and in 50 rounds for FedCO, saving 75% of communication rounds, while it requires 190 and 60 rounds for FedProx (5% saving) and CMFL (70% saving), respectively. Furthermore, in the FashionMNIST Non-IID data distribution scenario, the model converges to an accuracy of 90% in 200 rounds for the FedAvg algorithm, and in 186 rounds for FedProx, saving only 7%. In contrast, it requires 70 and 150 communication rounds for FedCO and CMFL, with savings of 65% and 25%, respectively. The experimental results on the CIFAR-10 data show that the model trained in the IID and Non-IID data settings need 200 rounds for FedAvg to reach 85% of the accuracy, while it requires 188 rounds for FedProx to reach 85% in IID, and more than 200 rounds in Non-IID to obtain target accuracy. On the other hand, FedCO and CMFL require 30 and 80 rounds, respectively, to converge under the IID data distribution scenario. Furthermore, within the Non-IID data distribution setting, the model converges to an accuracy of 85% in 200 rounds for the FedAvg, while it requires 60 and 160 communication rounds for FedCO and CMFL, respectively. Similarly, in the Non-IID data distribution setting, the FedCO communication costs are reduced to 82% with the MNIST data, 65% with the FashionMNIST data, and up to 70% with the CIFAR-10 dataset, compared to the FedAvg.

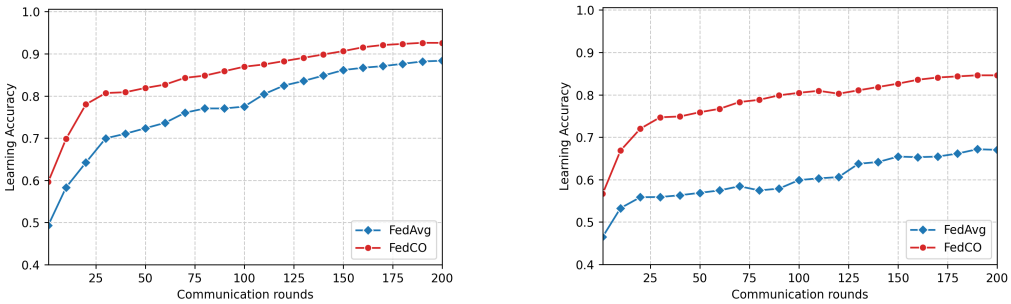
Although FedProx is considered to be an optimized version of FedAvg, we can observe from the results discussed above that FedProx behaves very similarly to FedAvg and shows only a slightly better performance than FedAvg in the conducted experiments. In addition, as we mentioned earlier, our FedCO algorithm can also be interpreted as an optimized version of FedAvg. Therefore, we further study these two algorithms (FedCO and FedAvg) by conducting experiments and benchmarking their performance on two datasets from the LEAF repository, namely FEMNIST and CelebA. Figure 8 shows the final accuracy scores after several rounds of communication for the FEMNIST dataset. Comparing the results produced by the two methods, it is evident that FedCO performs significantly better than FedAvg, on both the IID and Non-IID data scenarios. Specifically, FedCO ensures a higher accuracy than that of FedAvg within a smaller number of communication rounds. For example, in Figure 8a, FedCO can reach 90% in only 110 iterations, while the FedAvg never reaches that level within 200 iterations.



(**a**) (**b**)

Figure 8. Learning accuracy versus number of communication rounds for FEMNIST data. The top plot presents the results produced in the case of the IID data distribution scenario, while the bottom plot depicts the results generated in the case of the Non-IID data distribution scenario. (**a**) IID; (**b**) Non-IID.

Analyzing the results in Figure 9, we can observe the following: (1) FedCO consistently outperforms FedAvg in both data distribution scenarios; (2) FedCO generally achieves better accuracies than FedAvg in most cases (see Figure 9b), considering that both of them have been trained with only 200 rounds.



(**a**) (**b**)

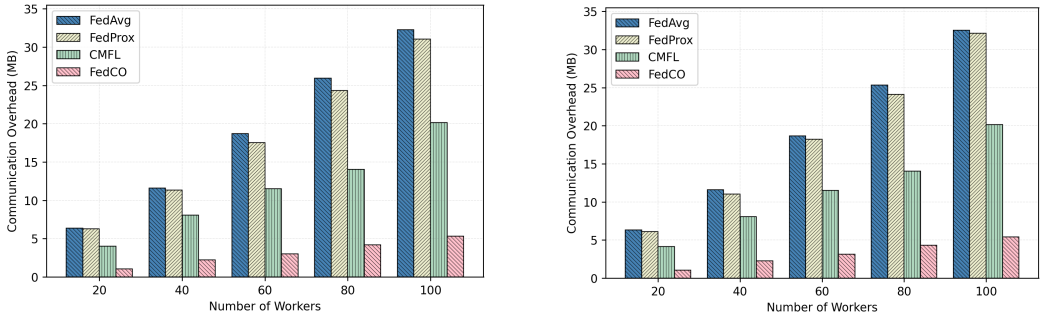
Figure 9. Learning accuracy versus number of communication rounds for CelebA data. The top plot presents the results produced in the case of the IID data distribution scenario, while the bottom plot depicts the results generated in case of the Non-IID data distribution scenario. (**a**) IID; (**b**) Non-IID.

Table 5 reports the number of communication rounds that the FedAvg and FedCO algorithms need in order to converge, for the considered datasets. Specifically, the target accuracy values are 70% for FEMNIST and 65% for CelebA, respectively. In addition, FedAvg is considered as the baseline.

Note that these results again verify the faster convergence of FedCO compared to that of FedAvg. Notice that we have also studied and compared FedProx and FedAvg on the same LEAF datasets, and they again have demonstrated very similar behaviors.

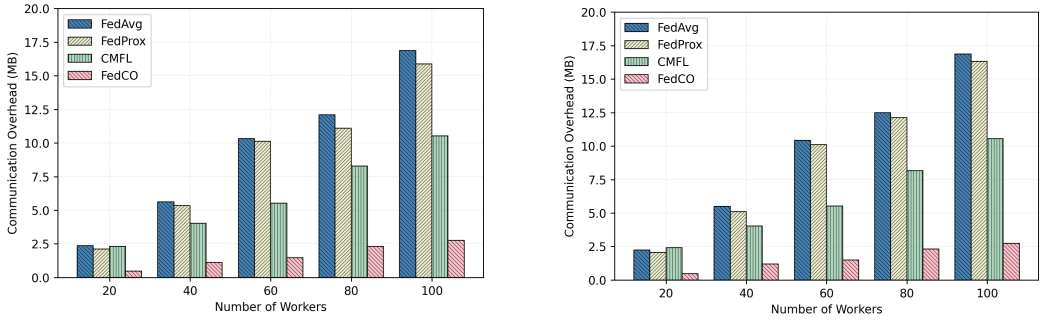
6.4. Communication Overhead Analysis

In this section, we compare the efficiencies of the two compared FL algorithms for 100 communication rounds with respect to different numbers of workers on the CIFAR-10 and the MNIST datasets, under the IID and Non-IID data distribution scenarios. The obtained results are reported in Figures 10 and 11, respectively. As one can notice, the FedCO algorithm has performed significantly better than the FedAvg, FedProx, and CMFL. The reader can also observe that the communication overhead increases linearly with the number of workers. Hence, to scale in a real scenario with thousands of workers, a FL algorithm should be capable of reducing the communication cost as much as possible, and reducing the number of rounds to converge, as with the proposed FedCO algorithm. Finally, the communication overhead in the IID and Non-IID cases is very close or identical. The results produced on the FashionMNIST dataset are similar to the other two datasets.



(**a**) (**b**)

Figure 10. The communication overhead for 100 rounds for the CIFAR-10 data. The top plot presents the results produced in the case of the IID data distribution scenario, while the bottom plot depicts the results generated in the case of the Non-IID data distribution scenario. (**a**) IID. (**b**) Non-IID.



(**a**) (**b**)

Figure 11. The communication overhead for 100 rounds for the MNIST data. The top plot presents the results produced in the case of the IID data distribution scenario, while the bottom plot depicts the results generated in the case of the Non-IID data distribution scenario. (**a**) IID. (**b**) Non-IID.

The communication cost savings for algorithm X (X = FedProx, CMFL, or FedCO) is computed as

$$1 - \frac{Communication\_overhead\_X}{Communication\_overhead\_FedAvg}.$$

As can be seen in Figure 10a,b, the FedCO costs on the CIFAR-10 IID data are 1 MB for 20 workers, which is a reduction in communication costs by 83% in comparison with FedAvg, while FedProx and CMFL are allowed to save only 12% and 36% in communication costs, respectively. In an experiment involving 100 workers on the CIFAR-10 dataset, FedAvg, FedProx, and CMFL exchange 32.5, 31.04, and 20 MB of data, while the proposed FedCO consumes only 5.4 MB, which means that FedCO reduces the communication overhead by 84% with respect to FedAvg, and CMFL reduces the communication overhead by 38%, while FedProx saves only 3% in communication costs.

Figure 11a,b report the communication costs under the MINIST IID and Non-IID data distribution scenarios, respectively. The trend is similar to the results of the CIFAR-10 data experiments. Both the IID and Non-IID data distribution settings confirm that our FedCO algorithm ensures a significantly smaller communication overhead in comparison with FedAvg, FedProx, and CMFL, by substantially reducing the required number of bytes exchanged. As can be noticed, FedCO allows a saving of between 80% and 85% with respect to FedAvg. In Figures 10 and 11, the communication costs increase linearly with the increasing number of workers for all of the compared algorithms. It is obvious that FedCO consistently outperforms FedAvg, FedProx, and CMFL in terms of reducing communication costs.

6.5. Threshold-Based Worker Selection

We also study scenarios in which we use an accuracy threshold to select the number of workers. The threshold is the specified cut-off of accuracy value for the selection of representatives of a cluster of workers. We select any worker where the local update ensures an accuracy of greater than or equal to the predefined threshold as a representative of a cluster. In this section, we report the results produced by testing four different threshold values for FedCO, namely 70%, 75%, 80%, and 85%. The network threshold for the selection of workers varies from bandwidth, transmission speed, or packet loss [48].

Table 6 reports the number of the top-ranked workers that the FedCO algorithm has selected to communicate with the server when the predefined threshold is met within 100 communication rounds.

Table 6. The number of selected representatives with respect to four different threshold values on the LEAF datasets CelebA (top) and FEMNIST (bottom) for 100 global rounds.

CelebA		
Threshold Accuracy	IID	Non-IID
≥70%	853	826
≥75%	673	515
≥80%	600	245
≥85%	257	226
FEMNIST		
Threshold Accuracy	IID	Non-IID
≥70%	912	844
≥75%	806	694
≥80%	730	604
≥85%	408	380

In the case of the CelebA data, the highest number of representatives has been selected when the accuracy of the local models is equal to or above 70%, namely 853 and 826 workers under IID and Non-IID, respectively. In the experiments conducted on the FEMNIST data, when the threshold of the local models was greater than or equal to 70%, 912 workers were selected as representatives for the IID scenario, and 844 workers for the Non-IID one. Similarly, these two values represent the highest numbers of selected workers. It is obvious from the number of representatives reported in Table 6 that the low threshold value implies the greater number of representatives to be selected for global training in FL and vice versa. Thus, we can observe that the proposed algorithm substantially reduces the accumulated communication overhead when FedCO selects only k representatives (i.e., one per cluster), rather than selecting a variable number of representatives based on a predefined threshold to train a global model.

Table 7 presents how many workers per round have been selected as representatives when various thresholds are applied for CelebA under the IID and Non-IID data scenarios, respectively.

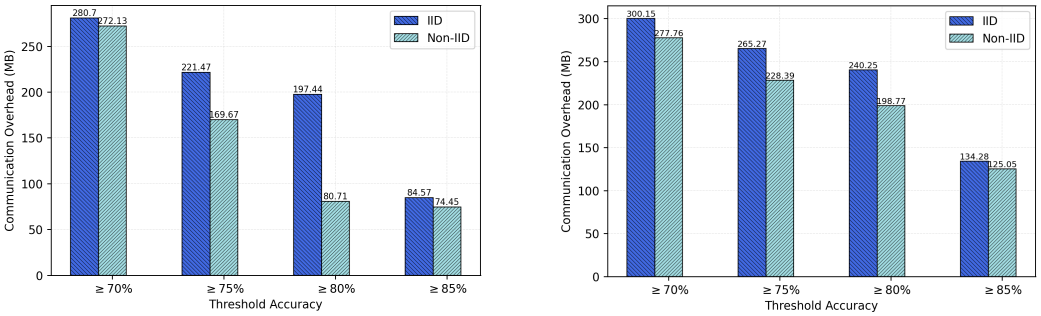
We can see that until 10 communication rounds, the FedCO selects only k representatives, since there are no local models where the accuracy has reached 70% at 10 rounds. Thus, the number of representatives increases from 10 to 97 at round 12 due to the selection of all the clusters' workers, ensuring an accuracy that is equal to or above the given threshold. Notice that there are 97 workers of different clusters that reach the value of accuracy of their local models of 70% or above. We can see that FedCO needs 30 rounds to have a number of workers whose accuracy is greater than or equal to 80% and to meet this condition under IID data. Furthermore, to meet the threshold of 85%, FedCO requires 100 rounds to have a number of workers (98) such that their accuracy of the local models meets this condition under IID. On the other hand, for Non-IID, FedCO never meets this

condition, since no local models have a accuracy value of higher than or equal to 85%; thus, FedCO selects only 36 workers to represent the different clusters.

Table 7. Total number of selected representatives when the given accuracy threshold is reached in CelebA dataset at different rounds.

	IID				Non-IID			
Round	**≥70%**	**≥75%**	**≥80%**	**≥85%**	**≥70%**	**≥75%**	**≥80%**	**≥85%**
1	10	10	10	10	10	10	10	10
10	10	12	16	14	14	16	18	16
12	97	13	16	16	14	18	22	18
20	102	103	20	20	104	26	25	24
30	105	104	95	18	104	30	28	24
40	102	108	98	20	110	32	30	26
50	106	108	100	22	112	96	32	28
60	106	110	104	24	114	99	34	28
70	107	111	106	26	116	102	36	32
80	108	114	108	22	120	104	38	35
90	110	116	110	26	118	106	38	34
100	112	116	112	98	122	108	97	36

Figure 12 provides communication overheads for various thresholds. It is obvious to the reader that a higher number of selected representatives implies a higher values of communication costs to the server.



(**a**) (**b**)

Figure 12. The communication overhead for 100 rounds for the two LEAF datasets. The top plot presents the results produced for CelebA dataset, while the bottom plot depicts the results generated on FEMNIST dataset. (**a**) CelebA; (**b**) FEMNIST.

The above results suggest that our proposed FedCO algorithm can substantially reduce the communication overhead by using a higher accuracy threshold. In general, FedCO can be considered as being robust to different application scenarios by being able to tune its parameters (e.g., the accuracy threshold or the number of top-ranked representatives per cluster) to find a trade-off between the application-specific resource constraints and the accuracy requirements.

7. Conclusions

This paper proposes a clustering-based FL approach, entitled Federated Learning using Clustering Optimization (FedCO). The proposed FedCO approach partially builds upon our previous work and extending further towards proposing a dynamic clustering scheme that improves global accuracy and that reduces the communication overhead in a Federated Learning context. The proposed approach dynamically identifies worker participants in each communication round by initially clustering the workers' local updates and selecting a representative from each cluster to communicate with the central server, thus minimizing the communication cost. The proposed FedCO method is evaluated

and benchmarked to three other state-of-the-art FL algorithms (FedAvg, FedProx, and CMFL) on five publicly available and widely exploited datasets for studying distributed ML algorithms. The experimental results have shown that the proposed FedCO algorithm significantly reduces communication rounds without sacrificing accuracy. In addition, the experimental evaluation has demonstrated that our FedCO algorithm outperforms the three other FL algorithms under the two studied data distribution scenarios. We have also shown that the FedCO algorithm can dynamically adapt the workers' partitioning at each communication round by relocating the representative workers and conducting the cluster splitting needed for the clustering improvement.

Our future plans include the enhancement of the FedCO approach through using other data distillation techniques; e.g., an interesting future direction could be made by applying computational topology methods for studying data topology and selecting representatives based on this. Another direction is the translation of the FedCO concept to unsupervised learning settings, i.e., developing a resource-efficient FL algorithm based on the unsupervised ML model.

Author Contributions: Conceptualization, A.A.A.-S., V.B. and E.C.; methodology, A.A.A.-S., V.B. and E.C.; software, A.A.A.-S.; validation, A.A.A.-S., V.B. and E.C.; formal analysis, A.A.A.-S. and V.B.; investigation, A.A.A.-S., V.B. and E.C.; data curation, A.A.A.-S.; writing—original draft preparation, A.A.A.-S., V.B. and E.C.; writing—review and editing, A.A.A.-S., V.B. and E.C.; visualization, A.A.A.-S.; supervision, V.B. and E.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The first author is supported by an Iraq Ministry of Higher Education and Scientific Research PhD Scholarship.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hatcher, W.G.; Yu, W. A Survey of Deep Learning: Platforms, Applications and Emerging Research Trends. *IEEE Access* **2018**, *6*, 24411–24432. [CrossRef]
2. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; The MIT Press: Cambridge, MA, USA, 2017.
3. Papernot, N.; McDaniel, P.; Sinha, A.; Wellman, M.P. Sok: Security and Privacy in Machine Learning. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK, 24–26 April 2018. [CrossRef]
4. Liang, F.; Hatcher, W.G.; Liao, W.; Gao, W.; Yu, W. Machine Learning for Security and the Internet of Things: The Good, the Bad, and the Ugly. *IEEE Access* **2019**, *7*, 158126–158147. [CrossRef]
5. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [CrossRef]
6. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. *Artif. Intell. Stat.* **2017**, *54*, 1273–1282.
7. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017. [CrossRef]
8. Sattler, F.; Wiedemann, S.; Muller, K.-R.; Samek, W. Sparse Binary Compression: Towards Distributed Deep Learning with Minimal Communication. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019. [CrossRef]
9. Al-Saedi, A.A.; Boeva, V.; Casalicchio, E. Reducing Communication Overhead of Federated Learning through Clustering Analysis. In Proceedings of the 2021 IEEE Symposium on Computers and Communications (ISCC), Athens, Greece, 5–8 September 2021. [CrossRef]
10. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated optimization in heterogeneous networks. *arXiv* **2021**, arXiv:1812.06127.
11. Wang, L.; Wang, W.; Li, B. CMFL: Mitigating Communication Overhead for Federated Learning. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–9 July 2019. [CrossRef]

12. Diao, E.; Ding, J.; Tarokh, V. HeteroFL: Computation and Communication Efficient Federated Learning for Heterogeneous Clients. *arXiv* **2021**, arXiv:2010.01264.
13. Reisizadeh, A.; Mokhtari, A.; Hassani, H.; Jadbabaie, A.; Pedarsani, R. FedPAQ: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization. *arXiv* **2020**, arXiv:1909.13014.
14. Sattler, F.; Muller, K.-R.; Samek, W. Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization under Privacy Constraints. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 3710–3722. [CrossRef] [PubMed]
15. Shlezinger, N.; Rini, S.; Eldar, Y.C. The Communication-Aware Clustered Federated Learning Problem. In Proceedings of the 2020 IEEE International Symposium on Information Theory (ISIT), Los Angeles, CA, USA, 21–26 June 2020. [CrossRef]
16. Kim, Y.; Hakim, E.A.; Haraldson, J.; Eriksson, H.; da Silva, J.M.; Fischione, C. Dynamic Clustering in Federated Learning. In Proceedings of the ICC 2021—IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021. [CrossRef]
17. Ghosh, A.; Chung, J.; Yin, D.; Ramchandran, K. An Efficient Framework for Clustered Federated Learning. *arXiv* **2020**, arXiv:2006.04088.
18. Ouyang, X.; Xie, Z.; Zhou, J.; Huang, J.; Xing, G. CLUSTERFL. In Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services, Virtual Event, WI, USA, 24 June–2 July 2021. [CrossRef]
19. Chen, Y.; Sun, X.; Jin, Y. Communication-Efficient Federated Deep Learning with Layerwise Asynchronous Model Update and Temporally Weighted Aggregation. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 4229–4238. [CrossRef]
20. Caldas, S.; Konecný, J.; McMahan, H.B.; Talwalkar, A.S. Expanding the Reach of Federated Learning by Reducing Client Resource Requirements. *arXiv* **2018**, arXiv:1812.07210.
21. Lin, Y.; Han, S.; Mao, H.; Wang, Y.; Dally, W. Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training. *arXiv* **2018**, arXiv:1712.01887.
22. Sattler, F.; Wiedemann, S.; Muller, K.-R.; Samek, W. Robust and Communication-Efficient Federated Learning from Non-I.i.d. Data. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *31*, 3400–3413. [CrossRef]
23. Vogels, T.; Karimireddy, S.P.; Jaggi, M. PowerSGD: Practical Low-Rank Gradient Compression for Distributed Optimization. In Proceedings of the NeurIPS 2019—Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
24. Asad, M.; Moustafa, A.; Ito, T. Fedopt: Towards Communication Efficiency and Privacy Preservation in Federated Learning. *Appl. Sci.* **2020**, *10*, 2864. [CrossRef]
25. Malekijoo, A.; Fadaeieslam, M.J.; Malekijou, H.; Homayounfar, M.; Alizadeh-Shabdiz, F.; Rawassizadeh, R. FEDZIP: A Compression Framework for Communication-Efficient Federated Learning. *arXiv* **2021**, arXiv:2102.01593.
26. Rothchild, D.; Panda, A.; Ullah, E.; Ivkin, N.; Stoica, I.; Braverman, V.; Gonzalez, J.E.; Arora, R. FetchSGD: Communication-Efficient Federated Learning with Sketching. *arXiv* **2020**, arXiv:2007.07682.
27. Xu, J.; Du, W.; Jin, Y.; He, W.; Cheng, R. Ternary Compression for Communication-Efficient Federated Learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *33*, 1162–1176. [CrossRef]
28. Chai, Z.; Chen, Y.; Anwar, A.; Zhao, L.; Cheng, Y.; Rangwala, H. FedAT: A high-performance and communication-efficient federated learning system with asynchronous tiers. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, St. Louis, MI, USA, 14–19 November 2021. [CrossRef]
29. Wu, X.; Liang, Z.; Wang, J. FedMed: A Federated Learning Framework for Language Modeling. *Sensors* **2020**, *20*, 4048. [CrossRef] [PubMed]
30. Asad, M.; Moustafa, A.; Aslam, M. CEEP-FL: A Comprehensive Approach for Communication Efficiency and Enhanced Privacy in Federated Learning. *Appl. Soft Comput.* **2021**, *104*, 107235. [CrossRef]
31. Nishio, T.; Yonetani, R. Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019. [CrossRef]
32. Park, S.; Suh, Y.; Lee, J. FedPSO: Federated Learning Using Particle Swarm Optimization to Reduce Communication Costs. *Sensors* **2021**, *21*, 600. [CrossRef]
33. Chen, Z.; Chong, K.F.E.; Quek, T.Q.S. Dynamic Attention-based Communication-Efficient Federated Learning. *arXiv* **2021**, arXiv:2108.05765.
34. Larsson, H.; Riaz, H.; Ickin, S. Automated Collaborator Selection for Federated Learning with Multi-Armed Bandit Agents. In Proceedings of the 4th FlexNets Workshop on Flexible Networks Artificial Intelligence Supported Network Flexibility and Agility, Virtual Event, 23 August 2021. [CrossRef]
35. Ji, S.; Pan, S.; Long, G.; Li, X.; Jiang, J.; Huang, Z. Learning Private Neural Language Modeling with Attentive Aggregation. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019. [CrossRef]
36. Ribero, M.; Vikalo, H. Communication-Efficient Federated Learning via Optimal Client Sampling. *arXiv* **2020**, arXiv:2007.15197.
37. Konečný, J.; McMahan, H.B.; Yu, F.X.; Richtarik, P.; Suresh, A.T.; Bacon, D. Federated Learning: Strategies for Improving Communication Efficiency. *arXiv* **2016**, arXiv:1610.05492.
38. MacQueen, J.B. Some methods for classification and analysis of multivariate observations. *Proc. Berkley Symp. Math. Stat. Probab.* **1967**, *1*, 281–297.

39. Rousseeuw, P.J. Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65. [CrossRef]
40. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
41. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv* **2017**, arXiv:1708.07747.
42. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. Available online: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (accessed on 18 April 2022).
43. Caldas, S.; Wu, P.; Li, T.; Konecný, J.; McMahan, H.B.; Smith, V.; Talwalkar, A.S. LEAF: A Benchmark for Federated Settings. *arXiv* **2018**, arXiv:1812.01097.
44. LeCun, Y.; Cortes, C. The Mnist Database of Handwritten Digits. Available online: https://www.lri.fr/~marc/Master2/MNIST_doc.pdf (accessed on 18 April 2022).
45. Kang, J.; Xiong, Z.; Niyato, D.; Xie, S.; Zhang, J. Incentive Mechanism for Reliable Federated Learning: A Joint Optimization Approach to Combining Reputation and Contract Theory. *IEEE Internet Things J.* **2019**, *6*, 10700–10714. [CrossRef]
46. Tran, N.H.; Bao, W.; Zomaya, A.; Nguyen, M.N.; Hong, C.S. Federated Learning over Wireless Networks: Optimization Model Design and Analysis. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications, Paris, France, 29 April 2019–2 May 2019. [CrossRef]
47. Konečný, J.; McMahan, H.B.; Ramage, D.; Richtarik, P. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv* **2016**, arXiv:1610.02527.
48. Zhou, P.; Fang, P.; Hui, P. Loss Tolerant Federated Learning. *arXiv* **2021**, arXiv:2105.03591.