FedRC: Representational Consistency Guided Model Uploading Mechanism for Asynchronous Federated Learning

Sheng Liu[1], Linlin You[1(✉)], and Yuren Zhou[2]

[1] School of Intelligent Systems Engineering, Sun Yat-Sen University, Shenzhen, China
liush235@mail2.sysu.edu.cn, youllin@mail.sysu.edu.cn
[2] Engineering Product Development, Singapore University of Technology and Design, Singapore, Singapore

Abstract. Recently, a novel distributed machine learning paradigm called Federated learning (FL) has caught the eyes of both academics and industries, as it can orchestrate substantial Internet of Things (IoT) devices as clients to learn a global model collaboratively and efficiently without sharing sensitive data. Moreover, while comparing the two modes of FL, i.e., synchronous FL (SFL) and asynchronous FL (AFL), AFL is more scalable and flexible to address the issue of over-fitting and the performance bottleneck caused by the stragglers. However, the data heterogeneity and high communication consumption issues faced by AFL still hamper its further applications and deployments in ubiquitous IoT. Motivated by this, we propose FedRC, a model uploading mechanism for AFL, guided by Representational Consistency (RC). As a layer-wise uploading method for Deep Neural Networks (DNNs), FedRC calculates simplified Representational Dissimilarity Vectors (RDVs) for each local layer and corresponding global layer, respectively, after the local training of each client, and then measures RCs based on the two RDVs to adaptively determine the uploading of model layers. According to the evaluation based on three standard datasets, compared with four state-of-the-art baselines (i.e., FedAvg, FedProx, FedAsync, and PartialNet), FedRC can boost model accuracy by 3.48%, save communication costs by 26.79%, and shorten transmission time by 44.14%, respectively.

Keywords: Federated Learning · Asynchronous Federated Learning · Deep Neural Networks (DNNs) · Model Uploading

1 Introduction

With the rapid development of IoT (Internet of Things), significant amounts of data are generated from massive smart devices such as Unmanned Aerial Vehicles, Intelligent Connected Vehicles, and medical robots [36,39]. Traditionally, these large amounts of data are transmitted to the data centers for utilization. However, this centralized

This research was funded by the National Natural Science Foundation of China (62002398) and the Collaborative Innovation Center for Transportation of Guangzhou (202206010056).

A. Zaslavsky et al. (Eds.): MobiQuitous 2023, LNICST 593, pp. 239–256, 2024.
https://doi.org/10.1007/978-3-031-63989-0_12

pattern is facing three main challenges, including large communication costs, privacy protection difficulties, and unbalanced computation loads between the center and edge devices.

In this context, a novel distributed learning paradigm, called Federated Learning (FL) [21], starts to attract more attention in both academic and industrial communities. In general, a FL system consists of one server and multiple clients to train a global model collaboratively and iteratively through multiple learning rounds. In each communication round, clients first receive the latest global model from the server, then perform local training based on their own datasets. Lastly, local model updates are uploaded to the server for aggregation to update the global model.

Since raw data are unexposed and edge resources are utilized, FL is more suitable to support ubiquitous IoT [16,40], and has been widely applied to various data-sensitive domains, e.g., to achieve the early-stage detection of dementia disease by sensors in smart homes [13], to perform image process and object detection by RSUs (Road-Side Units) in smart transportation [20], to identify suspicious transactions by banks in smart fintech [17], to recognize human activities by mobile phones in smart lives [5], etc.

Furthermore, according to the collaboration type of clients, FL can be divided into synchronous FL (SFL) and asynchronous FL (AFL) [1,38]. In SFL, a randomly selected or pre-selected subset of clients will participate in a certain round, and the server will execute the global aggregation process after all the participants have uploaded their local models successfully [21]. However, SFL has to wait for the stragglers (who may fail the training or communication process due to power-off or poor connection) in every round, which impedes the actual deployment of SFL. On the contrary, in AFL, clients can work individually and asynchronously, and the server will update the global model immediately once the pre-defined condition is satisfied (such as the maximum waiting time is reached or a certain number of updates are received) without unnecessary waiting [10,34].

Although AFL can improve scalability compared with SFL, it still faces two urgent challenges that affect the Quality of Service (QoS), namely:

- **Data Heterogeneity:** Since the sharing of raw data among clients is unallowed, local models of each client are trained and uploaded separately according to data collected with specific user preferences, which results in an intrinsic issue faced by AFL, i.e., strongly Non-IID (Non-Independent and Identically Distributed) data. Moreover, the staleness model caused by the asynchronous mode exacerbates the heterogeneous problem. Such an issue can degrade the learning performance of AFL significantly, as the representation of trained models can be partial [28].
- **Communication Resource Constraints:** First, the communication resources of edge clients are limited (restricted by hardware and software configurations) and may change over time (such as shifting between 5G and Wi-Fi). Second, each client needs to interact with the AFL server iteratively, which may become intrusive for other service co-running at the edge. The uncontrolled or unrestricted utilization of communication resources may easily become the bottleneck of IoT for the actual deployment. Finally, the communication burden of the AFL server can be large, since all clients are interacting with the AFL server frequently. Hence, training a global model with light communication costs is of significance to the development of AFL [7].

Accordingly, several solutions have been proposed to tackle these two challenges, such as adopting informative aggregation strategies to solve the data heterogeneity issue [30,37] and employing compression methods to reduce communication costs [18]. Specifically, to support the training of DNNs (Deep Neural Networks), a novel idea that can resolve the two issues simultaneously is to upload deep and shallow layers separately, as shallow layers contain fewer parameters while extracting more general features compared with deep layers [3,15,37]. However, currently, the fixed uploading frequency of different layers can not quantize the data heterogeneity level dynamically, and may also deteriorate the overall model performance. Such that, an adaptive mechanism, which measures the layer heterogeneity level caused by data heterogeneity for each client and only uploads those "fine" layers in each round to improve learning performance as well as save communication resources, is still missing.

To fill this gap, this paper proposes FedRC, a model uploading mechanism for AFL, guided by Representational Consistency (RC). To measure the layer heterogeneity adaptively, for each global layer and corresponding local layer, a simplified Representational Dissimilarity Vector (RDV) is calculated based on stimuli and pairwise distance functions after the local training and before the local model uploading. Then, the RC is computed according to the two RDVs, and utilized as the probability of uploading the local layer. Finally, after local layers are uploaded, each layer is aggregated at the server separately to form the layer-wise global model.

Accordingly, the main contributions of this paper are summarized as follows.

- This paper proposes a novel layer-wise uploading mechanism based on RC, called FedRC, to jointly address data heterogeneity and communication resource constraints issues faced by AFL.
- This paper designs a simplified version of RDM for the calculation of RC, named as RDV, to reduce the computation complexity significantly, and then, utilizes RC as the uploading probability of each layer in a round, to avoid the additional training of threshold.
- This paper evaluates FedRC based on three standard datasets (i.e., MNIST, FMNIST, and CIFAR-10), and compared with four state-of-the-art baselines (i.e., FedAvg, FedProx, FedAsync, and PartialNet), FedRC can improve model accuracy by 3.48%, reduce communication costs by 26.79%, and shorten transmission time by 44.14% simultaneously and respectively.

The remainder of this paper is organized as follows. First, Sect. 2 summarizes related work about AFL, model uploading methods, and similarity-based FL. Second, FedRC is presented and evaluated in Sect. 3 and Sect. 4, respectively. Finally, Sect. 5 concludes the work and sketches the future research directions.

2 Related Work

In this section, first, AFL is introduced briefly as the background; then current model uploading strategies for communication cost reduction are summarized; finally, the development of similarity in FL is reviewed. Note that the key related work is summarized in Table 1.

Table 1. The summary of key related work.

Group	Resolved Issue	Reference	Main contributions
Model Uploading	Update compression	[6]	Compressing gradients periodically and using local gradient tracking
		[35]	Introducing a self-learning quantization factor
		[29]	Combining NOMA with adaptive gradient quantization
		[26]	Presenting a Kashin compression method
	Topology optimization	[2]	Bridging SFL and AFL through tiers
		[9]	Introducing a multi-layer hybrid learning framework called fog learning
	Model split	[8]	Utilizing ordered dropout to extract lower footprint sub-models of DNNs
		[15]	Uploading the deep and shallow layers of DNNs with different frequencies
Similarity Analysis	Clustering	[27]	Grouping clients with similar data distributions by multitask learning
		[23]	Using pairwise client gradients to compute similarity for clustering
	RSA	[22]	Introducing RC to investigate the individual differences among DNNs
		[15]	Employing RC as adaptive weights to enhance global aggregation for AFL

2.1 Asynchronous Federated Learning

FedAsync is the first and the canonical AFL algorithm [34]. In FedAsync, the server will update the global model with a hyper-parameter related to the staleness in a weighted aggregation way after receiving a single local model from any client. ASO-Fed improves FedAsync by feature representation and dynamic learning rate, and supports online learning with continuous streaming local data [4].

However, such fully asynchronous methods have an unsteady learning process with model accuracy degraded, as the participant frequency and staleness among clients exit significant differences. Besides, significant transmission costs are also required because of the frequent interaction between clients and the server. Such that, FedSA [19] and SAFA [33] are proposed, which allow multiple clients instead of one client to participate in a global round and optimize the parameters of the learning process.

2.2 Model Uploading Methods

In general, the model uploading methods for communication cost reduction can be summarized in three ways, i.e., update compression, topology optimization, and model split, as illustrated in Table 1 "Model Uploading" group.

First, update compression refers to encoding and uploading updates by a smaller number of bits. Specifically, Haddadpour et al. [6] proposed FedCOMGATE: a federated averaging algorithm with compression and local gradient tracking for the heterogeneous setting; Xu et al. [35] designed FTTQ, which introduced a self-learning quantization factor to reduce the transmission of redundant parameters; Sun et al. [29] utilized adaptive gradient quantization and sparsification to facilitate the uploading scheme; Safaryan et al. [26] proposed Kashin compression: an unbiased compression method inspired by the Kashin representation of vectors.

Table 2. The summary of notations used in this paper.

Notation	Meaning
K	The total number of clients
K_t	The number of participants in the t^{th} round
w_k	The local model of client k
D_k	The local dataset of client k
$\|D_k\|$	The local data size of client k
w_t	The global model in the t^{th} round
$F_k(w_k)$	The local loss function of client k
η	The learning rate
L	The total number of layers
w_k^l	The l^{th} layer of local model w_k
K_t^l	The number of uploaded l^{th} layer in the t^{th} round
M	The stimulus set
h	The total number of stimuli
o_i^l	The output vector of the l^{th} layer on the stimulus i
P^l	The output dimension of the l^{th} layer
E	The dimension of RDV

Second, as for topology optimization, the novel idea is to design a hierarchical communication network, in which, each device only connects to a sub-center for relay uploading, and the server interacts with sub-centers periodically. E.g., Chai et al. [2] presented a tier-based mechanism named FedAT, which adopted a three-layer communication structure bridging SFL and AFL; and Hosseinalipour et al. [9] advocated fog learning as a multi-layer hybrid learning framework to achieve efficient communication based on flexible topology.

Finally, regarding the model split, the model parameters are split by various strategies for communication cost savings. E.g, Horvath et al. [8] proposed ordered dropout, which allowed each client to only train and upload a submodel of original DNNs; Chen et al. [3] and Liu et al. [15] both split the uploading schemes of deep and shallow layers of DNNs; and Wu et al. [32] presented FedKD based on adaptive mutual knowledge distillation, in which, a mentee model and a mentor model were learned from each other but only the parameters of mentee model were uploaded by clients.

2.3 Similarity Analysis in FL

As listed in Table 1 "Similarity Analysis" Group, it has been widely discussed in FL to discover divergent clients or boost the convergence speed [31]. Specifically, most related researches focus on the cluster of clients using different similarity measures, e.g., Sattler et al. [27] introduced clustered FL, which grouped clients with similar data

Table 3. The summary of abbreviations in this paper.

Abbr	Description
IoT	Internet of Things
FL	Federated Learning
SFL	Synchronous Federated Learning
AFL	Asynchronous Federated Learning
DNNs	Deep Neural Networks
QoS	Quality of Service
RSA	Representational Similarity Analysis
RDM	Representational Dissimilarity Matrix
RDV	Representational Dissimilarity Vector
RC	Representational Consistency
Non-IID	Non-Independent and Identically Distributed

distributions in multitask learning; and Palihawadana et al. presented FedSim [23], which clustered clients with similar gradients and utilized a metric called Privacy-preserving Non-IID Index to discover the relationship between similarity and Non-IID.

However, the Representational Similarity Analysis (RSA) [11,12,22], which can characterize the inner stimulus representation of DNNs, has not yet been explored in FL for the optimization of uploading. Even though Liu et al. [15] proposed Fed2A, which introduced RSA into AFL and computed the representational consistency as the adaptive weight in the global aggregation process, RSA is still not used in the model uploading process.

In general, AFL is more suitable for ubiquitous IoT because of its flexibility and scalability. However, the high communication consumptions impede its further application. Accordingly, update compression, topology optimization, and model split methods are widely studied to save communication resources. Whereas, side-effects such as performance dropping are also observed, and similarity-based methods,e.g., RSA, still need further exploration for the layer-wise uploading of DNNs, which are the key purposes of FedRC.

3 Methodology

As shown in Fig. 1, FedRC is proposed to optimize the uploading stage of AFL to save communication costs but without compromising QoS. In this section, first, the problem formulation is described; then the details of FedRC are presented; finally, the algorithm of FedRC is introduced. For ease of expression, the key notations and abbreviations used in this paper are summarized in Table 2 and Table 3, respectively.

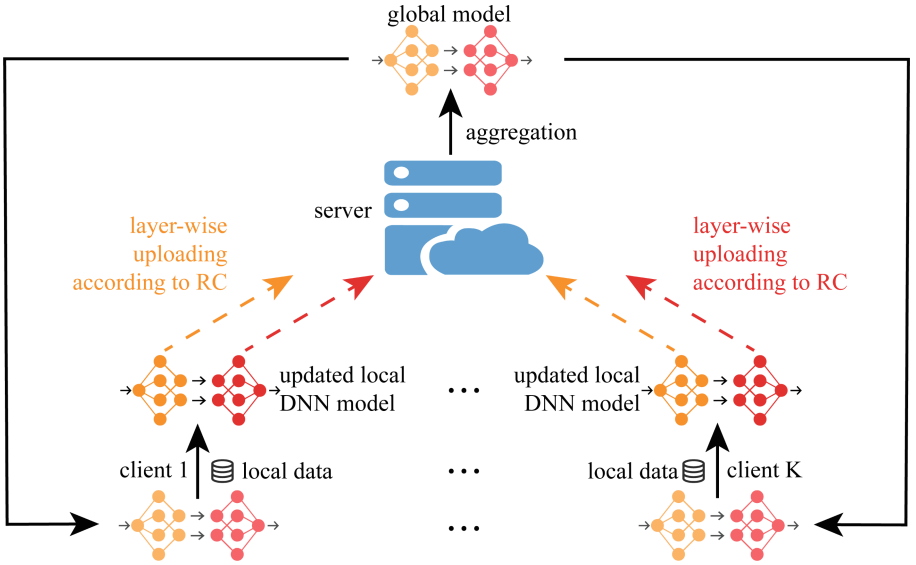

Fig. 1. The overall diagram of FedRC.

3.1 Problem Formulation

We consider an AFL system with total K clients, and each client k trains a local model w_k on its local dataset D_k with the size of $|D_k|$. Accordingly, the loss function of client k is defined as Formula 1,

$$F_k(w_k) = \frac{1}{|D_k|} \sum_{d_{k,i} \in D_k} f(d_{k,i}; w_k) \tag{1}$$

where D_k holds the training samples $d_{k,1}, d_{k,2}, ..., d_{k,|D_k|}$, and $f(\cdot)$ is a user-specified loss function, such as cross entropy and mean squared error.

After client k receives a global model w_{t_k} in the t_k^{th} global round, it conducts several local iterations for local update by formula 2,

$$w_k = w_{t_k} - \eta F'(w_{t_k}, D_k) \tag{2}$$

where η is the learning rate, and $F'(\cdot)$ is the gradients. Note that in AFL, for two different clients i and j, t_i is not necessarily equal to t_j, so as w_{t_i} and w_{t_j}.

Once the training of w_k is finished, w_k is uploaded to the server for global aggregation. Since in AFL, the aggregation process is unblocked, it can be triggered by a pre-defined condition, such as the maximum waiting time and the certain number of received local models of the server. Furthermore, the global updating can be executed by Formula 3,

$$w_{t+1} = \sum_{k=1}^{K_t} (\frac{|D_k|}{|D_t|} \times w_k) \tag{3}$$

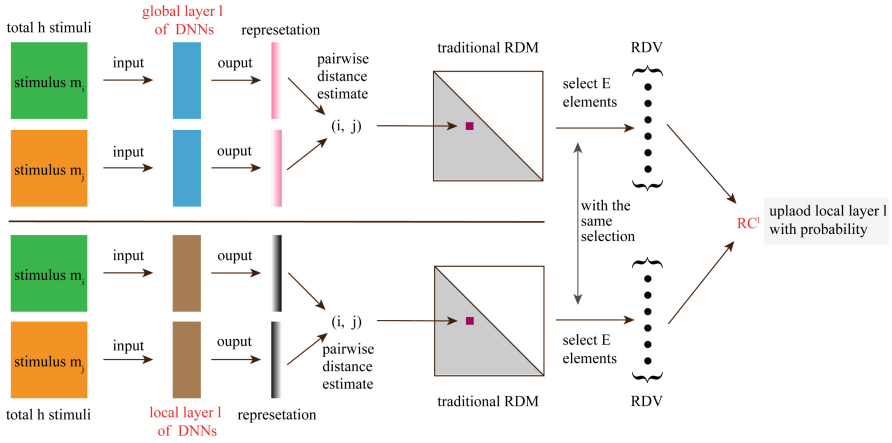

Fig. 2. The schematic diagram of FedRC for a client in a certain round.

where K_t and D_t are the number of participants and the total training samples in the t^{th} global round, respectively.

In this paper, we focus on the training of DNNs, hence w_k consists of multiple layers, which are denoted as L. Intuitively, the data heterogeneity brings model heterogeneity, i.e., layer heterogeneity. Moreover, deep layers (such as fully connected layers) contain many more parameters to learn ad hoc features compared with shallow layers (such as convolutional layers) [3]. Therefore, a layer-wise model uploading strategy is more suitable for AFL considering data heterogeneity and limited communication resources. In this context, we rewrite Formula 3 to Formula 4,

$$w_{t+1} = \sum_{l=1}^{L} \sum_{k=1}^{K_t^l} \left(\frac{|D_k|}{|D_t^l|} \times w_k^l \right) \tag{4}$$

where w_k^l is the l^{th} layer of w_k; K_t^l is the number of uploaded local layer l in the t^{th} round; and $D_t^l = \sum_{k=1}^{K_t^l} D_k$. Such that, the essential problem is determining whether to upload w_k^l in each round, which is solved by FedRC.

3.2 Details of FedRC

To measure the layer importance adaptively and guide the layer-wise model uploading strategy, Representational Similarity Analysis (RSA) [25] is adopted by FedRC. As shown in Fig. 2, given a set of stimuli $M = \{m_1, m_2, ..., m_h\}$, traditional RSA needs to compute a Representational Dissimilarity Matrix (RDM) with size $h \times h$ according to Formula 5,

$$RDM[i, j] = g(o_i^l, o_j^l), \quad i, j \leq h \tag{5}$$

where $o_i^l = o^l(m_i)$ denotes the output vector of l^{th} layer given stimula m_i; and $g(\cdot)$ measures the distance between two vectors, such as cosine distance (Formula 6), correlation distance (Formula 7), and Euclidean distance (Formula 8).

$$cos(o_i^l, o_j^l) = \frac{o_i^l \cdot o_j^l}{|o_i^l||o_j^l|} \tag{6}$$

$$cor(o_i^l, o_j^l) = 1 - \frac{Cov(o_i^l, o_j^l)}{\sqrt{D(o_i^l)}\sqrt{D(o_j^l)}} \tag{7}$$

$$euc(o_i^l, o_j^l) = \sqrt{\sum_{p=1}^{P^l}(o_i^l(p) - o_j^l(p))^2} \tag{8}$$

Note that the stimuli set M are prepared by the server according to public datasets and then are transmitted to all clients for computation. To reduce the computational complexity, we only randomly select and calculate E elements of the RDM to form a Representational Dissimilarity Vector (RDV). Such that, the simplified Representational Consistency (RC) between the global layer and local layer is defined according to Formula 9,

$$RC^l(RDV_{glo}^l, RDV_{loc}^l) = \rho(RDV_{glo}^l, RDV_{loc}^l)^2 \tag{9}$$

where ρ is the Pearson correlation coefficient.

A threshold can be designed to decide whether to upload w_k^l according to RC_k^l. However, such a threshold is a hard-coded uploading condition, and it requires strong experience or multiple experiments to set an appropriate threshold. To overcome this problem, FedRC introduces probability. Note that $0 \leq RC \leq 1$, hence RC^l can be used as the probability of uploading the l^{th} layer. To further regulate the probability to a proper scale, Min-Max normalization is employed according to Formula 10.

$$pro(w_k^l) = \frac{RC_k^l - min(RC^l)}{max(RC^l) - min(RC^l)} \tag{10}$$

3.3 Algorithm of FedRC

As shown in Algorithm 1, FedRC consists of two parts, namely:

Part 1: (in Each AFL Client). First, each AFL client k receives the global model w_{t_k} from the server in the t_k round. Second, client k clones w_{t_k} as the initialized local model w_k currently. Third, w_k is updated based on local dataset D_k. Fourth, the representational consistency RC_k^l is calculated for each layer. Finally, w_k^l is uploaded to the server-side with a probability computed by Formula 10.

Algorithm 1. The proposed FedRC algorithm

PART 1: Executed in each AFL client

for $k \leq K$ in parallel **do**
  Receiving the global model w_{t_k}
  $w_k = w_{t_k}$
  Training the local model w_k with D_k
  for layer $l \in L$ **do**
    Calculating RC_k^l by Formula 9
    Calculating $pro(w_k^l)$ by Formula 10
    Uploading w_k^l with probability $pro^(w_k^l)$
  end for
end for

PART 2: Executed in the AFL server

Thread 1: Receiving local models continuously
Thread 2: Aggregating local models periodically
while the aggregation condition is triggered **do**
  for $l \leq L$ **do**
    $w_{t+1}^l = \sum_{k=1}^{K_t^l} (\frac{|D_k|}{|D_t^l|} \times w_k^l)$
  end for
  $w_{t+1} = \sum_{l=1}^{L} w_{t+1}^l$
  Transmitting w_{t+1} to clients
  Increasing the global round t to $t+1$
end while

Part 2: (in the AFL Server). First, the AFL server receives local models from clients consistently. Second, once the aggregation condition is triggered, the server will update the global model layer by layer. Finally, the new global model w_{t+1} is transmitted to the client-side and the next global round begins.

Accordingly, the time complexity of the algorithm is $O(L \times h \times e_{max})$, where e_{max} denotes the maximum dimension of the output among the L layers. Although FedRC introduces additional computation for RC, the cost is actually negligible compared with the local training of DNNs.

In summary, as a layer-wise uploading AFL mechanism for the training of DNNs, FedRC introduces a simplified RC as the probability of transmission for each local layer at the client-side, while the server receives and aggregates local layer updates to improve the global model.

4 Experiment and Discussion

This section first describes the preparation of the experiment and then analyses the performance of FedRC comprehensively. Finally, the pros and cons of FedRC are further discussed based on the observations from experimental results.

4.1 Experiment Setting

We set up an AFL system comprising one server and 80 clients to evaluate the effectiveness and efficiency of FedRC. Note that the transmission time of clients ranges from 3 s to 30 s, and the AFL server aggregates the global model every 5 s. The system is developed based on TensorFlow libraries by using Python language, and we execute the experiment on a hardware device equipped with 4 NVIDIA GeForce RTX 3090 GPUs with 24 GB RAM based on Windows 10 Pro and CUDA v11.4.

Datasets and Models. The evaluations are conducted on three public datasets to train three corresponding CNN models. The default training settings are summarized in Table 4, and the details of datasets and corresponding model structures are listed as below:

- **MNIST**[1]: Modified National Institute of Standards and Technology dataset contains 60,000 handwritten digits for training and 10,000 for the test, with 10 labels and the image size of $28 \times 28 \times 1$. The CNN model has two stacked 5×5 convolution layers (32, 64 channels) followed by a 2×2 max pooling layer, two fully connected layers (128, 256 units) with relu activation function, and a softmax layer (10 units).
- **FMNIST**[2]: Fashion MNIST dataset has the same number of training samples, testing samples, and labels as MNIST. However, the images of FMNIST are clothes such as shirts and coats, which are more challenging to be classified correctly. The corresponding CNN model also shares the same structures as MNIST.
- **CIFAR-10**[3]: CIFAR-10 dataset comprises 60,000 $32 \times 32 \times 3$ colour images in 10 classes (such as the airplane and frog), with 6,000 images per class. There are 50,000 training images and 10,000 testing images. Since the task complexity is high, the CNN layers of CIFAR-10 have doubled channels or units compared with MNIST, and the other configurations are the same as MNIST.

Data Partition. To simulate a real-world IoT scenario, instead of IID data, we prepare Non-IID data based on the three datasets according to [3]. Specifically, for a dataset (i.e., MNIST, FMNIST, or CIFAR-10), each client is assigned with a data partition with training samples ranging from 500 to 1,500, and label numbers ranging from 1 to 6. Note that the fraction of each category in a partition is also imbalanced. As for the testing samples, they are untouched in the learning process and only used for evaluation.

[1] http://yann.lecun.com/exdb/mnist.
[2] https://github.com/zalandoresearch/fashion-mnist.
[3] https://www.cs.toronto.edu/~kriz/cifar.html.

Table 4. The summary of training setting.

Term	MNIST	FMNIST	CIFAR-10
Local epoch	5	5	5
Learning rate	0.003	0.001	0.001
Batch size	48	48	48
Optimizer	SGD	Adam	Adam

Benchmarks and Performance Metrics. Four state-of-the-art methods are adopted as the benchmarks for performance comparison, namely:

- **FedAvg** [21]: The canonical FL method proposed by Google, which selects participants randomly and aggregates local models averagely.
- **FedProx** [14]: It is a variant of FedAvg, which adds a proximal term μ on the local optimization function to address the data heterogeneity issue.
- **FedAsync** [34]: It is a popular AFL algorithm, which updates the global model immediately after receiving a local model according to a mixed hyperparameter α.
- **PartialNet** [24]: It is a communication-efficient FL algorithm, which only transmits the parameters of the largest layer of DNNs in each round.

To evaluate the learning performance, three metrics are used, i.e., 1) the maximum test accuracy during the training process according to Formula 11, where TP, TN, FP, and FN denote true positives, true negatives, false positives, false negatives, respectively; 2) the transmission time accumulated when the target accuracy (i.e., 90%, 85%, and 50% for MNIST, FMNIST, and CIFAR-10, respectively) is first reached; and 3) the total communication cost when the target accuracy is reached, which is calculated by Formula 12, where $n^r_{parameter}$ represents the size of uploaded parameters in the r^{th} round.

$$acc = \frac{TP + TN}{TP + FP + FN + TN} \tag{11}$$

$$cost_{target} = \sum_{r=1}^{r_{target}} \left(\frac{4 \times n^r_{parameter}}{1024 \times 1024}\right) \tag{12}$$

4.2 Performance Evaluation

First, stimuli sets are needed to calculate RC for the uploading probability of model layers. Specifically, for each dataset, 100 images (10 categories $\times$ 10 stimuli) from the testing samples are prepared as the stimuli, and the dimension of RDV E is set as 100. Compared with the original RDM, which consists of $4,900$ unique elements, RDV can significantly save computational resources. Since there are three different distance functions as described by Formula 6-8, RedRC has three variants, namely FedRC-cos/cor/euc, and they will be evaluated and compared respectively.

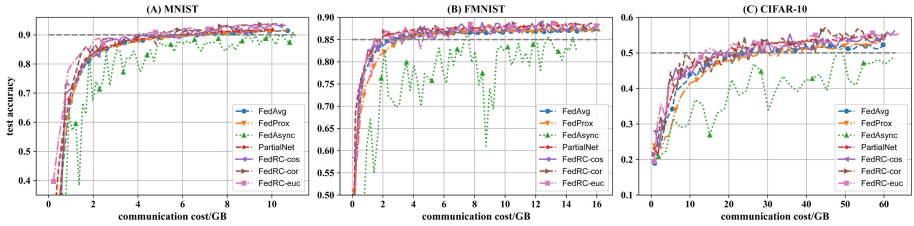

Fig. 3. Test accuracy vs. communication cost in (A) MNIST; (B) FMNIST; and (C) CIFAR-10.

Table 5. The experiment results of baselines and FedRC.

Method	Maximum test accuracy			Transmission time (in seconds)		
	MNIST	FMNIST	CIFAR-10	MNIST	FMNIST	CIFAR-10
FedAvg	91.48%	87.57%	52.35%	3623	1416	2148
FedProx	91.67%	87.34%	53.70%	3562	1615	2680
FedAsync	91.78%	87.60%	54.76%	**534**	**458**	**545**
PartialNet	**91.87%**	**88.77%**	**54.97%**	3247	979	1954
FedRC-cos	95.47%	88.84%	56.26%	475	270	325
FedRC-cor	**95.56%**	**88.99%**	57.67%	505	**185**	**290**
FedRC-euc	95.21%	88.89%	**58.36%**	**395**	195	330

1) Evaluation of Test Accuracy

First, as shown in Table 5, among the four baselines, PartialNet achieves the highest test accuracy on all three datasets, i.e., 91.87% on MNIST, 88.77% on FMNIST, and 54.97% on CIFAR-10, which suggests the advantages of layer-wise FL algorithm.

Second, all the three FedRC variants can outperform the best baseline PartialNet on the three datasets. Specifically, the best variant for each dataset, namely FedRC-cor, FedRC-cor, and FedRC-euc, can improve accuracy by about 4.02%, 0.25%, and 6.17% on MNIST, FMNIST, and CIFAR-10, respectively.

Third, the three FedRC variants have similar test accuracies on MNIST and FMNIST, while the differences on CIFAR-10 are slightly large, which indicates that complicated tasks are more sensitive to the distance function.

In summary, the saving on communication costs of FedRC will not deteriorate the model performance, and surprisingly, an average improvement of 3.48% on the test accuracy is observed. The results show that FedRC can filter the redundancy of local models caused by data heterogeneity, and only uploads and aggregates more useful local layers to form a better global model.

2) Evaluation of Communication Efficiency

First, as illustrated in Fig. 3, FedAsync needs much more communication resources for model training compared with other baselines (i.e., FedAvg, FedProx, and PartialNet), and the curve of FedAsync is much more vibrant, which suggests that general AFL algorithms sacrifice communication efficiency and learning stability for stragglers.

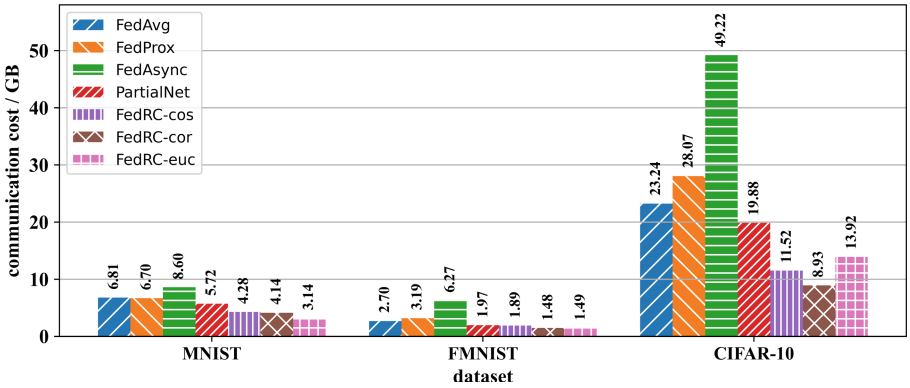

Fig. 4. The communication cost of FedRC compared with the baselines in the three datasets.

Second, when consuming the same communication cost, FedRC can not only improve accuracy significantly compared with FedAsync, but also boost model performance slightly compared with FedAvg, FedProx, and PartialNet.

Finally, as shown in Fig. 4, FedRC spends less communication cost to reach target accuracy compared with the baselines, namely:

- **For MNIST**: FedRC-cos, FedRC-cor, and FedRC-euc only need 4.28, 4.14, and 3.14 GB to reach 90% accuracy respectively, and an average reduction of 32.63% is observed compared with the best baseline PartialNet (5.72 GB);
- **For FMNIST**: The average consumption of the three FedRC variants is 1.62 GB, while the best baseline PartialNet needs 1.97 GB to get an accuracy of 85%. Hence, the reduction of cost is 17.77%;
- **For CIFAR-10**: Compared with the best performance of the four baselines (i.e., PartialNet 23.24 GB to reach 50%), the maximum, minimum, and average improvements in communication efficiency of FedRC variants are 55.08%, 29.98%, and 42.37%, respectively.

In summary, since FedRC adopts the layer-wise uploading approach based on RC-based uploading probability, the improvement of FedRC on communication efficiency is significant, and an average cost reduction of 26.79% against the best baseline of the three datasets is observed. Note that all three distance functions are favourable, and the stability of FedRC is not affected by the reduction of uploaded parameters.

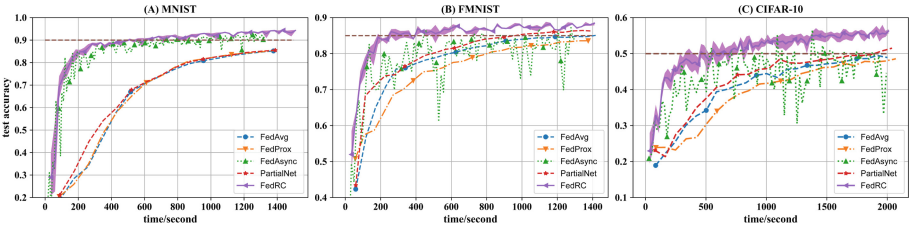

Fig. 5. Test accuracy vs. transmission time in (A) MNIST; (B) FMNIST; and (C) CIFAR-10.

3) Evaluation of Time Efficiency
First, as summarized in Table 5, among the four baselines, FedAsync presents the shortest transmission time (534/458/545 s) to reach the target accuracy on all three datasets (MNIST/FMNIST/CIFAR-10), which indicates the high time efficiency of AFL algorithms.

Second, all three FedRC variants can reach the target accuracy faster than FedAsync. Specifically, 1) FedRC-cor performs the best on both FMNIST and CIFAR-10, as it only needs 185 s and 290 s to reach the target accuracies of 85.00% and 50.00% with a reduction of 59.61% and 46.79% compared with FedAsync, respectively; 2) FedRC-euc is the fastest to reach the accuracy of 90% on MNIST, and a boost on the learning speed of 26.03% against the best baseline is observed.

Finally, as illustrated in Fig. 5, the three FedRC variants share a similar time-accuracy curve, which outperforms FedAsync as well as FedAvg, FedProx, and PartialNet. Note that FedRC does not vibrate sharply compared with FedAsync, and can gain accuracy improvement quickly compared with FedAvg, FedProx, and PartialNet.

In summary, FedRC inherits and enhances the advantages in time efficiency of AFL methods, and in the meanwhile, can maintain a more stable learning curve.

4.3 Discussion

First, four main advantages of FedRC are observed from the results: 1) improving the model accuracy; 2) saving communication costs; 3) shortening the transmission time; 4) maintaining a stable learning curve.

Second, as for the comparisons among different distance functions, we discover the following three interesting observations: 1) all three distance functions are effective as they share similar performance improvements and learning curves; 2) the complicated dataset may be more sensitive to the distance function, as its accuracies under different distances are more diverse; 3) the correction distance has higher compatibility for FedRC and can be the default method, as FedRC-cor outperforms FedRC-cos and FedRC-euc slightly in most cases in terms of the test accuracy and transmission time. Intuitively, all three distance functions can measure the similarity between two representational vectors, and the correlation distance may be more suitable for complex datasets, which generally require high-dimensional representations.

Finally, although the learning curve of FedRC is much more stable compared with FedAsync, it still vibrates slightly compared with FedAvg and FedProx, which may be due to the absence of utilizing temporal attributes to improve the aggregation process.

5 Conclusion

In this paper, we present FedRC, a layer-wise model uploading mechanism for asynchronous federated learning. Based on the representational dissimilarity vector, representational consistency is calculated as the layer uploading probability, since it can capture the similarity of stimulus representation between the local layer and the corresponding global layer.

A comprehensive evaluation using three datasets (i.e., MNIST, FMNIST, and CIFAR-10) demonstrates that FedRC outperforms four state-of-the-art baselines (i.e., FedAvg, FedProx, FedAsync, and PartialNet) by about 3.48%, 26.79%, and 44.14% in terms of learning performance, communication efficiency, and transmission time efficiency, respectively. The results prove that representational consistency can indeed be utilized as the uploading probability to transmit more valuable layer parameters with communication cost reduced and model accuracy improved.

However, a slight performance fluctuation during the learning is observed in FedRC when compared to FedAvg and FedProx, and additional computation is required for the calculation of RC, which indicate that FedRC can be further improved. In the future, we will focus on the following directions: 1) the RDV will be further optimized to enhance the representation level without increasing computational resources; 2) complicated DNNs with more layers will be utilized in the experiments to further evaluate FedRC and reveal the relationship between layer number and representational consistency; and 3) layer-wise aggregation strategies will be designed to improve learning stability.

References

1. Bonawitz, K., et al.: Towards federated learning at scale: system design. Proc. Mach. Learn. Syst. **1**, 374–388 (2019)
2. 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, pp. 1–16 (2021)
3. 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. **31**(10), 4229–4238 (2020). https://doi.org/10.1109/TNNLS.2019.2953131
4. Chen, Y., Ning, Y., Slawski, M., Rangwala, H.: Asynchronous online federated learning for edge devices with non-iid data. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 15–24. IEEE (2020)
5. Ek, S., Portet, F., Lalanda, P., Vega, G.: Evaluation and comparison of federated learning algorithms for human activity recognition on smartphones. Pervasive Mob. Comput. **87**, 101714 (2022). https://doi.org/10.1016/j.pmcj.2022.101714
6. Haddadpour, F., Kamani, M.M., Mokhtari, A., Mahdavi, M.: Federated learning with compression: unified analysis and sharp guarantees. In: International Conference on Artificial Intelligence and Statistics, pp. 2350–2358. PMLR (2021)
7. Herabad, M.G.: Communication-efficient semi-synchronous hierarchical federated learning with balanced training in heterogeneous iot edge environments. Internet Things **21**, 100642 (2023). https://doi.org/10.1016/j.iot.2022.100642
8. Horvath, S., Laskaridis, S., Almeida, M., Leontiadis, I., Venieris, S., Lane, N.: Fjord: fair and accurate federated learning under heterogeneous targets with ordered dropout. Adv. Neural. Inf. Process. Syst. **34**, 12876–12889 (2021)
9. Hosseinalipour, S., Brinton, C.G., Aggarwal, V., Dai, H., Chiang, M.: From federated to fog learning: distributed machine learning over heterogeneous wireless networks. IEEE Commun. Mag. **58**(12), 41–47 (2020). https://doi.org/10.1109/MCOM.001.2000410
10. Hu, C.H., Chen, Z., Larsson, E.G.: Scheduling and aggregation design for asynchronous federated learning over wireless networks. IEEE J. Sel. Areas Commun. **41**(4), 874–886 (2023). https://doi.org/10.1109/JSAC.2023.3242719

11. Kornblith, S., Norouzi, M., Lee, H., Hinton, G.: Similarity of neural network representations revisited. In: International Conference on Machine Learning, pp. 3519–3529. PMLR (2019)
12. Kriegeskorte, N., Mur, M., Bandettini, P.: Representational similarity analysis - connecting the branches of systems neuroscience. Front. Syst. Neurosci. **2** (2008). https://doi.org/10.3389/neuro.06.004.2008
13. Li, J., et al.: A federated learning based privacy-preserving smart healthcare system. IEEE Trans. Ind. Inf. **18**(3), 2021–2031 (2022). https://doi.org/10.1109/TII.2021.3098010
14. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proc. Mach. Learn. Syst. **2**, 429–450 (2020)
15. Liu, S., Chen, Q., You, L.: Fed2a: federated learning mechanism in asynchronous and adaptive modes. Electronics **11**(9), 1393 (2022). https://doi.org/10.3390/electronics11091393
16. Liu, S., Qu, H., Chen, Q., Jian, W., Liu, R., You, L.: Afmeta: asynchronous federated meta-learning with temporally weighted aggregation. In: 2022 IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Meta), pp. 641–648 (2022). https://doi.org/10.1109/SmartWorld-UIC-ATC-ScalCom-DigitalTwin-PriComp-Metaverse56740.2022.00100
17. Liu, Y., et al.: Fedvision: an online visual object detection platform powered by federated learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 13172–13179 (2020)
18. Lu, X., Liao, Y., Lio, P., Hui, P.: Privacy-preserving asynchronous federated learning mechanism for edge network computing. IEEE Access **8**, 48970–48981 (2020). https://doi.org/10.1109/ACCESS.2020.2978082
19. Ma, Q., Xu, Y., Xu, H., Jiang, Z., Huang, L., Huang, H.: FEDSA: a semi-asynchronous federated learning mechanism in heterogeneous edge computing. IEEE J. Sel. Areas Commun. **39**(12), 3654–3672 (2021)
20. Manias, D.M., Shami, A.: Making a case for federated learning in the internet of vehicles and intelligent transportation systems. IEEE Network **35**(3), 88–94 (2021). https://doi.org/10.1109/MNET.011.2000552
21. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence and Statistics, pp. 1273–1282. PMLR (2017)
22. Mehrer, J., Spoerer, C.J., Kriegeskorte, N., Kietzmann, T.C.: Individual differences among deep neural network models. Nat. Commun. **11**(1), 1–12 (2020). https://doi.org/10.1038/s41467-020-19632-w
23. Palihawadana, C., Wiratunga, N., Wijekoon, A., Kalutarage, H.: Fedsim: similarity guided model aggregation for federated learning. Neurocomputing **483**, 432–445 (2022). https://doi.org/10.1016/j.neucom.2021.08.141
24. Paragliola, G., Coronato, A.: Definition of a novel federated learning approach to reduce communication costs. Expert Syst. Appl. **189**, 116109 (2022). https://doi.org/10.1016/j.eswa.2021.116109
25. Raghu, M., Gilmer, J., Yosinski, J., Sohl-Dickstein, J.: SVCCA: singular vector canonical correlation analysis for deep learning dynamics and interpretability. In: 31st International Conference on Neural Information Processing Systems, pp. 6078–6087 (2017)
26. Safaryan, M., Shulgin, E., Richtárik, P.: Uncertainty principle for communication compression in distributed and federated learning and the search for an optimal compressor. Inf. Inference J. IMA **11**(2), 557–580 (2022). https://doi.org/10.1093/imaiai/iaab006
27. Sattler, F., Müller, K.R., Samek, W.: Clustered federated learning: model-agnostic distributed multitask optimization under privacy constraints. IEEE Trans. Neural Netw. Learn. Syst. **32**(8), 3710–3722 (2021). https://doi.org/10.1109/TNNLS.2020.3015958

28. Sattler, F., Wiedemann, S., Müller, K.R., Samek, W.: Robust and communication-efficient federated learning from non-i.i.d. data. IEEE Trans. Neural Netw. Learn. Syst. **31**(9), 3400–3413 (2020). https://doi.org/10.1109/TNNLS.2019.2944481
29. Sun, H., Ma, X., Hu, R.Q.: Adaptive federated learning with gradient compression in uplink noma. IEEE Trans. Veh. Technol. **69**(12), 16325–16329 (2020). https://doi.org/10.1109/TVT.2020.3027306
30. Wang, X., Li, R., Wang, C., Li, X., Taleb, T., Leung, V.C.M.: Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching. IEEE J. Sel. Areas Commun. **39**(1), 154–169 (2021). https://doi.org/10.1109/JSAC.2020.3036946
31. Wang, Y., Wolfrath, J., Sreekumar, N., Kumar, D., Chandra, A.: Accelerated training via device similarity in federated learning. In: Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking, pp. 31–36 (2021)
32. Wu, C., Wu, F., Lyu, L., Huang, Y., Xie, X.: Communication-efficient federated learning via knowledge distillation. Nat. Commun. **13**(1), 1–8 (2022). https://doi.org/10.1038/s41467-022-29763-x
33. Wu, W., He, L., Lin, W., Mao, R., Maple, C., Jarvis, S.: SAFA: a semi-asynchronous protocol for fast federated learning with low overhead. IEEE Trans. Comput. **70**(5), 655–668 (2021). https://doi.org/10.1109/TC.2020.2994391
34. Xie, C., Koyejo, S., Gupta, I.: Asynchronous federated optimization. arXiv preprint arXiv:1903.03934 (2019)
35. Xu, J., Du, W., Jin, Y., He, W., Cheng, R.: Ternary compression for communication-efficient federated learning. IEEE Trans. Neural Netw. Learn. Syst. **33**(3), 1162–1176 (2022). https://doi.org/10.1109/TNNLS.2020.3041185
36. You, L., He, J., Wang, W., Cai, M.: Autonomous transportation systems and services enabled by the next-generation network. IEEE Netw. **36**(3), 66–72 (2022). https://doi.org/10.1109/MNET.006.2100542
37. You, L., Liu, S., Chang, Y., Yuen, C.: A triple-step asynchronous federated learning mechanism for client activation, interaction optimization, and aggregation enhancement. IEEE Internet Things J. **9**(23), 24199–24211 (2022). https://doi.org/10.1109/JIOT.2022.3188556
38. You, L., Liu, S., Zuo, B., Yuen, C., Niyato, D., Vincent Poor, H.: Federated and asynchronized learning for autonomous and intelligent things. IEEE Netw. (2023). https://doi.org/10.1109/MNET.2023.3321519
39. You, L., Tunçer, B., Zhu, R., Xing, H., Yuen, C.: A synergetic orchestration of objects, data, and services to enable smart cities. IEEE Internet Things J. **6**(6), 10496–10507 (2019). https://doi.org/10.1109/JIOT.2019.2939496
40. Zhang, Y., Suleiman, B., Alibasa, M.J.: Fedgroup: a federated learning approach for anomaly detection in iot environments. In: Longfei, S., Bodhi, P. (eds.) MobiQuitous 2022. LNCS, vol. 492, pp. 121–132. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-34776-4_7