Communication-Loss Trade-Off in Federated Learning: A Distributed Client Selection Algorithm

Minoo Hosseinzadeh[1], Nathaniel Hudson[1], Sam Heshmati[2], and Hana Khamfroush[1]

[1]Department of Computer Science, University of Kentucky
[2]Department of Marketing & Supply Chain, University of Kentucky

2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC) | 978-1-6654-3161-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CCNC49033.2022.9700601

***Abstract*—Mass data generation occurring in the Internet-of-Things (IoT) requires processing to extract meaningful information. Deep learning is commonly used to perform such processing. However, due to the sensitive nature of these data, it is important to consider data privacy. As such, federated learning (FL) has been proposed to address this issue. FL pushes training to the client devices and tasks a central server with aggregating collected model weights to update a global model. However, the transmission of these model weights can be costly, gradually. The trade-off between communicating model weights for aggregation and the loss provided by the global model remains an open problem. In this work, we cast this trade-off problem of client selection in FL as an optimization problem. We then design a Distributed Client Selection (DCS) algorithm that allows client devices to decide to participate in aggregation in hopes of minimizing overall communication cost — while maintaining low loss. We evaluate the performance of our proposed client selection algorithm against standard FL and a state-of-the-art client selection algorithm, called Power-of-Choice (PoC), using CIFAR-10, FMNIST, and MNIST datasets. Our experimental results confirm that our DCS algorithm is able to closely match the loss provided by the standard FL and PoC, while on average reducing the overall communication cost by nearly 32.67% and 44.71% in comparison to standard FL and PoC, respectively.**

***Index Terms*—Internet-of-Things, Federated Learning, Client Selection, Distributed Learning, Machine Learning**

I. Introduction

The Internet-of-Things (IoT) includes a vast number of connected devices that produce a huge amount of data that need to be processed. Deep Learning (DL) [1] has been shown as a viable tool to process and analyze IoT-generated data. For performing DL across IoT devices, the most straight-forward approach is to rely on a central server to collect raw data from the IoT devices to then perform model training on the central server. However, this approach causes privacy issues regarding IoT devices' data as the raw data of users are shared with a central entity. Further, there is expensive communication cost due to the large distance between the IoT devices and a remote central server — also the sheer size of IoT-generated data. The recent advent of Edge Computing [2] can provide improved privacy, reduced overall latency, and other advantages due to physical proximity to the IoT devices. While, the volume of produced data is large enough to elicit smart decisions regarding how resources are spent [3]–[6].

A recently proposed solution for these challenges is the framework of *federated learning* (FL) [7]–[9]. Under the framework of FL, we consider a central server and K client devices each with their own data. The central server initializes a global DL model (e.g., some deep neural network for image classification) with parameters set w and shares it with $K \cdot C$ clients where $C \in [0, 1]$; so that they each host a local copy of the global model. Then, each client device performs *local training* using its own dataset for the now locally-hosted DL model. Now, clients just send their locally-updated models to the server to update the global model rather than submitting the raw data. Then, in the *federated aggregation* phase, the central server computes weighted average of the received parameters to update the global model for redistribution in the next round. This process occurs over a number of rounds until the global model's loss converges. Since no raw data produced by the clients are transmitted, there is an immediate layer of privacy provided under the framework of FL [7]–[9]. In this paper, we refer to the FL presented in [9] as *Standard FL.*

Although FL provides privacy and reduces the total communication cost of the clients by not communicating their raw data [2], clients still have to consume energy and spend communication capacity to upload their locally-updated models. Additionally, the initially-proposed FL [7]–[9] considers random client selection. However, recent papers have shown that *biased* client selection (i.e. client selection considering different criteria such as clients' data) can accelerate the error convergence [10]–[13]. Although several papers considered the communication cost associated with uploading the locally-trained models by proposing smart client selection algorithms [14], there is still a gap in fully understanding the trade-off between communication cost and the final loss value of global model in FL for IoT environments. In this paper, we solve the client selection problem while focusing on making a trade-off between the total communication cost of sending the locally-updated parameters from clients to the server and the final test loss of the global model. To the best of our knowledge, this is the first work which considers the trade-off between the overall communication cost and loss of the global model in the federated learning environment. We highlight the main contributions of the paper as following:

- We cast the trade-off between communication cost and model loss in the FL environment as an optimization problem where communication between the clients and the server might be costly due to limited capacities of the client devices.

- We propose a Distributed Client Selection (DCS) algorithm to solve the proposed optimization problem where the clients independently decide whether to send their locally-updated models to the server.
- We show that our algorithm outperforms standard FL and another client selection baseline presented in [12] (namely, PoC explained in §V), in terms of the provided communication-loss trade-off over 3 datasets.

II. Related Work

Initially proposed by Konečnỳ et al. in [7], *federated learning* (FL) performs distributed learning across devices with their own unique datasets while providing an immediate layer of privacy. The `FedAvg` aggregation algorithm proposed by McMahan et al. in [9] demonstrates promising training loss/accuracy under the FL framework even in the face of Non-IID data distributions across client devices. One of main challenges in the FL environments is *Client Selection* which has been addressed in two different phases: *(i)* the *training phase* where a set of clients are selected to perform *local training*, and *(ii)* the *aggregation phase* where a set of clients' locally-updated models are selected to contribute to federated aggregation. In [9], the former phase is done randomly, and *all* selected clients from the first phase participate in aggregation phase. In the following, we briefly describe some of recent works in this area of client selection for FL.

Client Selection for Training Phase. Several papers address the client selection problem for training phase to reduce the overall communication cost [14]–[19]. Papers focused on either wireless channel properties while minimizing the model loss [15], or maximizing the number of clients in each round to minimize the total number of communication rounds [17], or minimizing the convergence time [14], [16], [18], [19] to reduce the communication cost. Our work departs from these works by focusing on client selection to reduce the overall communication cost for aggregation phase to not losing any client's update due to different factors such as latency.

Client Selection for Aggregation Phase. Several papers address the client selection problem at the aggregation phase to reduce the overall communication cost [10]–[13], [20], [21]. Papers focus on different goals, such as bandwidth allocation [20], and minimizing communication cost [11]. Authors in [13] propose a client selection algorithm to maximize the final accuracy of the global model where the server tests clients' locally updated models and chooses the top-R clients with the highest accuracy for aggregation. Cho et al. in [12] propose a centralized client selection algorithm in which clients with higher loss over their local dataset are selected for aggregation. Authors in [10] proposed an algorithm to minimize the total number of convergence rounds whilst guaranteeing the model convergence, while our focus is on making a trade-off between the model convergence and the overall communication cost.

Though, client selection in both phases has been studied, to their best of knowledge past works did not explicitly consider the trade-off between communication and model performance (via test loss). Thus, we study client selection during the aggregation phase while considering the trade-off between these two. We do this by running small batches of local testing over a globally-shared small validation dataset (SVD). We then let the clients independently decide to participate in aggregation upon performing this local testing.

III. Problem Formulation

System Setup. We consider a system with one server and K clients such that clients are connected to the server through a wireless connection. Given the mentioned system, FL employs a global loss function $F(w, \mathcal{D})$ over dataset $\mathcal{D}$ to facilitate the learning process where at each round t ($t \in \{1, \cdots, \tau\}$), a random subset of clients of size $\max(1, \lfloor C \cdot K \rfloor)$ are selected to train the model locally, where $C \in (0, 1]$ indicates the selection rate parameter. In round t, we denote the set of selected clients to train as $\mathcal{S}_t$ and clients participating in aggregation by sending their locally-updated models to the server as $\mathcal{V}_t$.

Problem Formulation. Here, we present a mathematical formulation for making a trade-off between the communication cost due to sending the locally-updated models from clients to the server and the final loss value in FL. We design an optimization model to minimize the the overall communication cost whilst guaranteeing a specific level of final loss value. For the sake of this work, we focus on the communication cost caused by sending locally-updated models from clients to the server. We assume that the cost of broadcasting the global model from server to the clients is negligible [9], [11]–[13]. To minimize the communication cost, we make a decision for selecting a portion of clients to send their locally-updated models whilst providing a competitive level of final loss.

We introduce the *Communication-Aware Client Selection Problem* (CASP), which aims to minimize the overall communication cost on the client side such that the trained FL model achieves a pre-defined desired threshold of loss value, l_d. Note that l_d represents a pre-defined/given desired goal loss value. Each client k has a local training dataset $\mathcal{D}_k$ to train the model locally. At round t, the server sends the global model with parameters w_t, to the selected clients, $\mathcal{S}_t$, to train the model locally and update the local weights. Then, a subset $\mathcal{V}_t$ of these selected clients, $\mathcal{S}_t$, send their locally-updated weights to the server where $\mathcal{V}_t \subseteq \mathcal{S}_t$. We define $c_t^k \in (0, 1]$ as the known (normalized) communication cost of sending the updated model parameters from client k to the server at round t. The normalized communication cost may represent various aspects (e.g., energy, bandwidth) related to data transmission without needing to change the problem definition. Once the server receives the local weights, it updates the global weights of round $t+1$, w_{t+1}, as:

$$w_{t+1} \triangleq \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k, \tag{1}$$

where n_k is the number of data items of client k and $n \triangleq \sum_{k=1}^{K} n_k$ is the total number of data items across all clients. w_{t+1}^k is the model parameters of the local model hosted by client k in time-step $t+1$, and w_{t+1} is the parameters of global model at $t+1$. For the CASP, we define a decision variable $x_t^k = 1$ if and only if client k is selected to send its locally-updated model to the server at round t, and 0 otherwise. We define $\mathcal{V}_t \triangleq \{k \in \mathcal{S}_t | x_t^k = 1\}$ as the set of selected clients to send their updated models to the server. For training, let local model weights w_{t+1}^k for each client k to be the following:

$$w_{t+1}^k = \begin{cases} w_t - \eta \nabla F(w_t, \mathcal{B}_k) & \text{if } k \in \mathcal{V}_t \\ w_t & \text{otherwise} \end{cases} \tag{2}$$

where η is the fixed learning rate, $F(w_t, \mathcal{D}_k)$ is the loss of using weights w_t on client's local data $\mathcal{D}_t$, $\nabla \ell(w_t; \mathcal{D}_k)$ is average gradient from the loss function. In Eq. (2), the first case corresponds with client k participating in aggregation in time-step t; the second case is for when client k does not participate in aggregation so that aggregation uses the global model's weights w_t for that client. The global loss function of round t is denoted by $F(w_t, \mathcal{D})$ where $\mathcal{D}$ represents the test dataset available at the server.[1] Thus, CASP is defined below:

$$\text{min:} \sum_{k=1}^{K} \sum_{t=1}^{\tau} c_t^k x_t^k \tag{3}$$

$$\text{s.t.:} \; F(w_\tau, \mathcal{D}) \le l_d \tag{3a}$$

$$\sum_{k=1}^{K} x_t^k \ge 1 \quad \forall t \in \{1, \cdots, \tau\} \tag{3b}$$

$$x_t^k \in \{0,1\} \qquad \forall k \in \{1, \cdots, K\}, t \in \{1, \cdots, \tau\} \tag{3c}$$

The objective in Eq. (3) minimizes the total communication cost of FL from clients' side over τ rounds by choosing the minimum number of clients participating in the aggregation phase, where τ is a large integer value to guarantee that the problem is feasible. (3a) ensures the test loss obtained by the global model with weights of the last round, w_τ, is less than or equal to the desired goal loss, l_d. (3b) ensures each round has at least 1 participating client in aggregation phase. (3c) states that the decision variable is binary.

A. Problem Complexity

Robustly proving this problem is NP-*hard* is not straightforward due to the intrinsic reliance on the optimization algorithms (e.g., stochastic gradient descent) to perform the local model training to update local model weights w_{t+1}^k $(\forall k, t)$. As can be seen in Eq. (1), w_{t+1} depends on the values of w_{t+1}^k $(\forall k, t)$. Because the values of w_{t+1}^k $(\forall k, t)$ are not predetermined and are updated throughout the decision-making process based on our binary decision variable, x_t^k, it is not obvious to approach the complexity of this problem as a standard ILP problem. Instead, the provided optimization framework for the CASP is used to give a sense of what the optimal solution for our problem would look like — where the minimum number of clients are selected to meet our loss threshold of l_d. However, because the model architectures of interest for FL (e.g., neural networks with many layers comprised of multiple neurons/units) consist of a composition of many convex/concave functions [23], the resulting architectures are often non-convex or non-concave. Global maximization of non-convex and global minimization of non-concave functions are both NP-*hard* problems [24]. This suggests that the CASP is likely a hard problem. However, a robust NP-hardness proof is beyond the scope of this work due to the the fact that decisions related to x_t^k depend on optimization of non-conxev/non-concave functions. Therefore, exact approaches to CASP, based on a commercial integer programming solver, are unlikely to be applicable in a realistic client selection problem due to their size from a performance perspective. With that in mind, next section presents an adaptive distributed client selection algorithm to generate efficient solution for CASP.

[1]Note that the our problem definition is general and works for any loss function (e.g., cross-entropy, Hinge, mean squared error) [22].

IV. Proposed Algorithm

As discussed, to generate fast, high quality solutions, employing heuristic approach is inevitable. Based on our initial observations, after few rounds, local model test loss of some clients will be very similar to the global model test loss while some other clients may still show large test loss computed over a given small validation dataset. The large loss value on clients side means that they still have some data items which the global model does not perform well on them. Based on this observation, we believe it is possible to collect locally-updated parameters from fewer than the total set of $\max(1, \lfloor C \cdot K \rfloor)$ selected clients. If correct, this would reduce communication cost. Thus, we propose a distributed client selection algorithm that clients make real-time decisions on whether they should participate in aggregation phase in a given round or not. The goal is to reach the desired final loss value l_d.

Small Validation Dataset (SVD). We consider a Small Validation Dataset (SVD) that the server broadcasts to the clients only once before starting the first round. The SVD has a uniform distribution of labels for consistency. The SVD is used for quick evaluations of local model performance in terms of loss. We assume both the server and the clients have a copy of SVD. The server selects a small, static subset of test dataset (i.e., SVD$\subset \mathcal{D}$) and sends it to all clients willing to participate in FL process before the first round.

The pseudocode for our algorithm, *Distributed Client Selection* (DCS), is presented in Algorithm 1. Here, we describe the intuition behind our algorithm. In each round, our algorithm first finds the test loss of the global model (l_t) using the small validation dataset, $F(w_t,\text{SVD})$ (line 4). Then, it randomly selects a set of clients with size $|\mathcal{S}_t| = \max(1, \lfloor C \cdot K \rfloor)$ to build set $\mathcal{S}_t$ (line 6). The server then distributes the global model with parameters w_t alongside the loss of global model computed at round t over SVD, l_t, to the selected clients $\mathcal{S}_t$. Then, the `clientside` procedure will start (line 9). The clients in set $\mathcal{S}_t$ start to train the global model using their own unique local training data (line 3 of `clientside` procedure). Then, each client $i \in \mathcal{S}_t$ tests its locally-updated model over SVD and finds the loss l_t^i. If it holds that $l_t^i \ge l_t$ for client i, then client i will send its locally-updated model parameter w_t^i to the server (line 4-6 of `clientside` procedure). When the

Algorithm 1: Proposed DCS

```
1  Procedure serverside()
       Input  : $SVD$, $K$, $C$, $l_d$
       Output: Trained global model parameters $w_\tau$
2      Initialize global model with parameters $w_0$;
3      Send small validation dataset $SVD$ to all clients
         $\forall k \in K$;
4      Initialize $l_t \leftarrow$F($w_0, SVD$);
5      while $l_t \geq l_d$ do
6          $\mathcal{S}_t \leftarrow$ subset of $\max(1, \lfloor C \cdot K \rfloor)$ random clients;
7          $\mathcal{V}_t \leftarrow \{\}$;
8          for each selected client $i \in \mathcal{S}_t$ in parallel do
9              ans $\leftarrow$ clientside($i$,$w_t, l_t$);
10             if $ans \neq 0$ then $\mathcal{V}_t \leftarrow \mathcal{V}_t \cup \{ans\}$ ;
11         if $\mathcal{V}_t = \varnothing$ then $\forall i \in \mathcal{S}_t$ receive updates;
12         $w_{t+1} \leftarrow$ updated global parameters via Eq. (1).
             $l_t \leftarrow F(w_t, SVD)$;
13     return $w_\tau$;
1  Procedure clientside($i, w_t, l_t$)
       Input  : Client ID $i$, Server parameters $w_t$, $l_t$
       Output: Local model parameters $w_t^i$ if participating in
                aggregation phase.
2      if $w \neq null$ then
3          $w_t^i \leftarrow$ update weights with local training data;
4          $l_t^i = F(w_t^i, SVD)$;
5      if $l_t^i \geq l_t$ then return $w_t^i$ ;
6      else return 0 ;
```

server receives a new update from a client, it adds that client to the set $\mathcal{V}_t$ to use for aggregation (line 10). If the server does not receive any update from any client, it will ask all clients to send their locally-updated models (line 11). Finally, the server aggregates the received locally-updated models using Eq. 1 with model parameters equal to Eq. 2 (line 12). This repeats until the global test loss meets a desired loss value l_d (line 6).

V. Experimental Results

A. Experimental Design

Baseline Algorithms. We chose 2 algorithms as baselines, namely Standard FL [9] (FedAvg), and Power-of-Choice (PoC) [12]. The PoC selects $d \geq \max(1, C \cdot K)$ number of clients in each round for training phase. After training using their local dataset, they send back their updates to the server. Finally, server chooses $\max(1, C \cdot K)$ number of clients with higher loss value over their local dataset for aggregation phase [12]. We run PoC's algorithm for $d = K(C + 0.1)$.

Model Architectures & Data. We implement our DCS algorithm in Python 3 and consider deep learning models using the PyTorch API. We consider 3 datasets to evaluate DCS in terms of communication cost-loss trade-off: MNIST [25], Fashion-MNIST (FMNIST) [26], and CIFAR-10 [27]. For our experiments, we consider a simple convolutional neural network (CNN) model for training over each dataset assuming the data is distributed among clients in a Non-Independent, Identically Distributed (Non-IID) fashion. Our CNNs incorporate the ReLU activation function and we use *cross-entropy* as the loss function, $F(w, \mathcal{D})$ [22]. Code for this work and the considered CNNs can be found at [28]. Both the FMNIST and the MNIST data consist of 60,000 and 10,000 images for training and testing, respectively. For the Non-IID data distribution for both MNIST and FMNIST, data are sorted by label then divided into 200 contiguous shards of size 300, and all the $K = 100$ clients are given 2 random shards each, similar to [9]. For the CIFAR-10 dataset which consists of 50,000 training images and 10,000 testing images, the Non-IID setting consists of 200 contiguous shards each consisting of 250 images. Additionally, we use Eq. (1) for aggregation.

Parameters. We consider $K = 100$ clients. Unless otherwise stated, we uniformly sample a random value from the range $c_t^k \in (0, 1]$ $(\forall k, t)$ for each client k's communication cost. For simplicity, we assume the c_t^k is fixed during all training rounds for each client (i.e., $c_{t'}^k = c_t^k$ $(\forall t, t' \in \tau)$). We perform numerical experiments to investigate the effect of distribution of c_t^k in our experiments in the results. Each client has a local minibatch size of $B = 10$ and a number of local epochs of $E = 5$ for local training. We set the learning rate $\eta = 0.001$. We initialize a random set of initial model parameters w_0 for the global DNN model and fix the random seed for all experiments. The test data batch size is 128. We set the l_d to be equal to the test loss value of global model trained by the Standard FL l_{FL} at a given number of rounds for each dataset plus an small error margin ε (i.e., $l_d = l_{FL} + \varepsilon$). Note that l_d can be set to any other value in the real world and the reason that we set it to the test loss value of global model trained by the Standard FL plus an small error margin ε is to have a fair comparison against the Standard FL. For MNIST and FMNIST, we set the number of FL rounds to 100 and for CIFAR-10 we set the number of FL rounds to 200. For the PoC baseline algorithm we define the number of rounds similarly. Note that for our proposed DCS algorithm the number of rounds is not fixed and the algorithm stops whenever the desired goal loss l_d is achieved. We set $\varepsilon = 0.01$. We repeat each experiment for different fractions of users, namely, $C = 0.1, 0.3, 0.5, 0.7, 0.9$.

SVD Selection. We randomly select 200 images from the test datasets of each considered dataset with an equal number of images per labels to avoid bias toward a specific label.

B. Results & Discussion

We investigate the final loss value of the global model by testing it over the large test dataset which is assumed to be available on the central server. We also investigate the effect of error margin ε on the total communication cost. We define the Total Communication Cost (TCC) as the sum of the communication cost of all clients spent over all rounds which they participated in the aggregation phase via communicating their locally-updated models, thus $TCC = \sum_{k=1}^{K} \sum_{t=1}^{\tau} c_t^k x_t^k$. Additionally, we define the Communication Cost Ratio (CCR) as $CCR = \frac{\text{DCS TCC}}{\text{FedAvg TCC}}$. Thus, the lower CCR, the more significant the provided gain w.r.t. communication cost reduction.

Communication Cost-loss Trade-off. Here, we discuss the communication-cost trade-off that DCS provides compared to the baseline. We tested some scenarios when $C < 0.5$, and

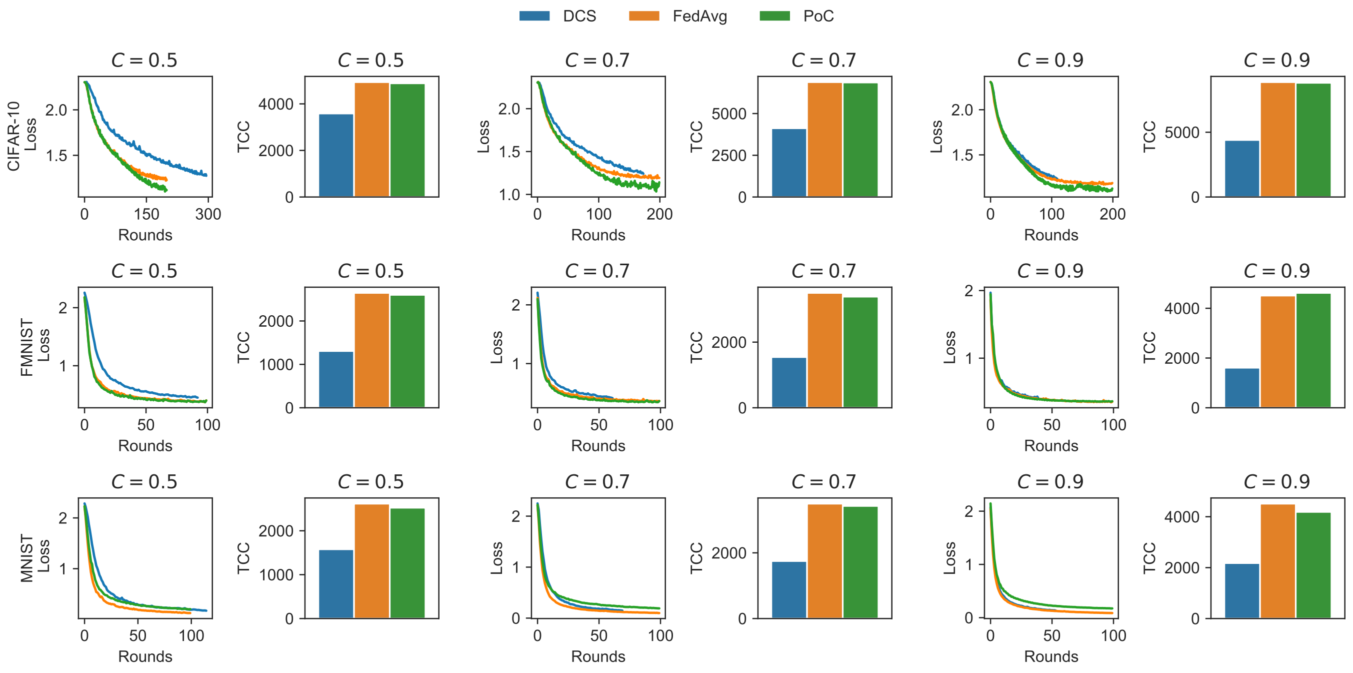

Fig. 1: Total Communication Cost (TCC) vs. Final Global Model Loss Over Large Test Dataset

$\varepsilon = 0.01$ and the improvement of communication cost was not significant compared to the baseline. Because, in these scenarios we do not have enough freedom to choose best clients in each round when $C < 0.5$ and $K = 100$. Therefore, we focus on $C \geq 0.5$ for the results section. As shown in Fig 1, our algorithm outperforms both the FedAvg and PoC algorithms in terms of TCC minimization whilst still providing almost equal amount of final test loss value of global model with the error margin $\varepsilon = 0.01$. In most of cases, DCS stops sooner than the baseline algorithms as shown in Fig 1. In few cases such as in CIFAR-10, when $C = 0.5$, the number of total rounds of our algorithm is slightly more than the baselines. However, the TCC value of DCS is still less than that of the baseline algorithms; so that, it still beats the baseline algorithms in terms of total communication cost reduction.

Moreover, we define *Communication Cost Reduction Ratio* (CCRR) as $CCRR = 1 - \frac{\text{DCS TCC}}{\text{Baseline TCC}}$ for each baseline showing how much communication cost reduction we gained compared to that baseline. In average, the CCRR of three datasets is equal to 27.39%, 37.06%, 33.58% for MNIST, FMNIST, and CIFAR-10 in comparison to FedAvg, and 63.76%, 39.89%, and 30.49% in comparison to PoC, respectively.

Impact of ε on CCR. For some applications (e.g., IoT environments where the communication is costly), the communication cost is much more important than a small error w.r.t. the goal loss error ε. We investigate the effect of ε on CCR to find the impact of error margin on the communication cost-loss trade-off when $C = 0.5$ over all three datasets. As presented in Table I, when we increase the error margin, the communication cost will decrease for DCS compared to the FedAvg. The reason is that DCS algorithm stops whenever it meets the goal loss plus the error margin. Hence, when the error margin is large, DCS stops sooner and will not run for next rounds. This process provides the trade-off between the communication cost and the final loss value of the global model. Hence, our algorithm can perform well in terms of making a better trade-off between overall communication cost and the final loss value for these kinds of applications.

TABLE I: Comm. Cost Ratio (CCR) vs. ε Error Margin.

	DCS's Comm. Cost Ratio		
Error Threshold	CIFAR-10	FMNIST	MNIST
$\varepsilon = 0.01$	0.7270	0.4940	0.6029
$\varepsilon = 0.02$	0.7270	0.4940	0.6029
$\varepsilon = 0.03$	0.7270	0.4940	0.6029
$\varepsilon = 0.04$	0.7270	0.4940	0.6029
$\varepsilon = 0.05$	0.7251	0.4881	0.5972
$\varepsilon = 0.06$	0.7016	0.4374	0.5671
$\varepsilon = 0.07$	0.6601	0.4009	0.5156
$\varepsilon = 0.08$	0.6483	0.3742	0.4780
$\varepsilon = 0.09$	0.6304	0.3629	0.4780
$\varepsilon = 0.10$	0.6279	0.3261	0.4045

Impact of Clients Communication Cost Distribution on Total Communication Cost. IoT devices are usually heterogeneous in terms of communication capacity. To investigate the effect of communication cost distribution of users on DCS, we adjust the distribution of communication costs of the clients that participate in aggregation phase. We use normal distribution to generate the communication cost of each client and investigate the total communication cost of each algorithm over 1000 Monte Carlo runs for different values of mean and variance and set $\varepsilon = 0.01$. Our results in Fig. 2 for C values of 0.1 and 0.9 imply that the proposed DCS algorithm performs

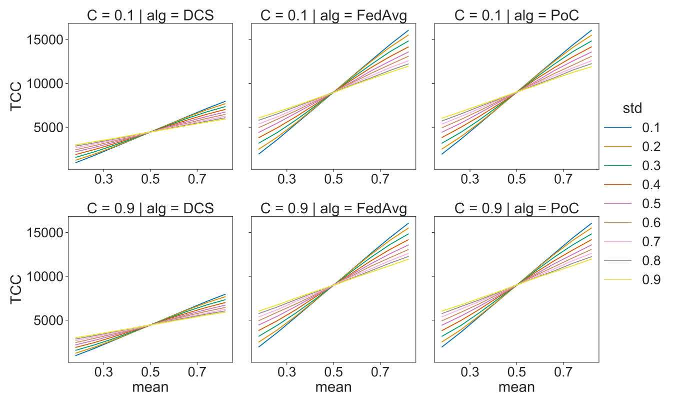

Fig. 2: Total Communication Cost (TCC) vs. Clients' Communication Cost Distribution

better in terms of total communication cost reduction in all cases and considering non-uniform communication cost of the clients in the network. Even when we change the std of normal distribution to 0.9, our algorithm still performs well.

The Communication Overhead of DCS. DCS algorithm has no communication overhead aside from sending the SVD from the server to the clients only once before the first round (SVD is fixed during the FL rounds). Since the focus of this paper is the communication cost of clients, sending SVD from server to clients does not affect clients' communication cost.

VI. Conclusion

This paper studies client selection for federated learning (FL) while considering the trade-off between the final loss of global model and the communication cost on the client side to upload their locally-updated models. We cast this problem as an optimization problem and propose a novel distributed algorithm, which we call DCS, where clients decide to send their locally-updated model weights based on test loss using a small shared dataset provided by the server. We compare our proposed algorithm with standard federated learning and a state-of-the-art client selection algorithm, called Power-of-Choice (PoC), in terms of final loss and the overall communication cost. Our results demonstrate that our DCS algorithm is able to reduce the communication cost associated with uploading the locally-updated model weights for aggregation in FL by more than 32.67% and 44.71% on average in comparison to standard FL and PoC, respectively.

Acknowledgment

This work is funded by research grant provided by the National Science Foundation (NSF) under the grant number 1948387.

References

[1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[2] N. Hudson, M. J. Hossain, M. Hosseinzadeh, H. Khamfroush, M. Rahnamay-Naeini, and N. Ghani, "A framework for edge intelligent smart distribution grids via federated learning," in *2021 International Conference on Computer Communications and Networks (ICCCN)*.

[3] X. Zhao, M. Hosseinzadeh, N. Hudson, H. Khamfroush, and D. E. Lucani, "Improving accuracy-latency trade-off of edge-cloud computation offloading for deep learning services," in *IEEE Globecom Workshop on Edge Learning over 5G Networks and Beyond*, 2020.

[4] M. Hosseinzadeh, A. Wachal, H. Khamfroush, and D. E. Lucani, "Optimal accuracy-time trade-off for deep learning services in edge computing systems," in *IEEE International Conference on Communications*, 2021.

[5] N. Hudson, H. Khamfroush, and D. E. Lucani, "QoS-aware placement of deep learning services on the edge with multiple service implementations," in *2021 IEEE ICCCN*, 2021, pp. 1–8.

[6] M. Hosseinzadeh, N. Hudson, X. Zhao, H. Khamfroush, and D. E. Lucani, "Joint compression and offloading decisions for deep learning services in 3-tier edge systems," in *IEEE DySPAN*, 2021.

[7] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[8] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

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

[10] W. Luping, W. Wei, and L. Bo, "Cmfl: Mitigating communication overhead for federated learning," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*.

[11] M. Ribero and H. Vikalo, "Communication-efficient federated learning via optimal client sampling," *arXiv preprint arXiv:2007.15197*, 2020.

[12] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv*.

[13] I. Mohammed, S. Tabatabai, A. Al-Fuqaha, F. El Bouanani, J. Qadir, B. Qolomany, and M. Guizani, "Budgeted online selection of candidate iot clients to participate in federated learning," *IEEE IoT Journal*, 2020.

[14] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *IEEE ICC*, 2019.

[15] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. on Wireless Communications*, 2020.

[16] T. Huang, W. Lin, W. Wu, L. He, K. Li, and A. Zomaya, "An efficiency-boosting client selection scheme for federated learning with fairness guarantee," *IEEE Trans. on Parallel and Distributed Systems*, 2020.

[17] S. AbdulRahman, H. Tout, A. Mourad, and C. Talhi, "Fedmccs: multi-criteria client selection model for optimal IoT federated learning," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4723–4735, 2020.

[18] N. Yoshida, T. Nishio, M. Morikura, and K. Yamamoto, "Mab-based client selection for federated learning with uncertain resources in mobile networks," in *2020 IEEE Globecom Workshops (GC Wkshps)*.

[19] Y. J. Cho, S. Gupta, G. Joshi, and O. Yağan, "Bandit-based communication-efficient client selection strategies for federated learning," in *2020 54th Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2020, pp. 1066–1069.

[20] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Trans. on Wireless Communications*, 2020.

[21] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. A. Jarvis, "Safa: a semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Transactions on Computers*, 2020.

[22] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT.

[23] P. Jain and P. Kar, "Non-convex optimization for machine learning," *arXiv preprint arXiv:1712.07897*, 2017.

[24] K. G. Murty and S. N. Kabadi, "Some NP-complete problems in quadratic and nonlinear programming," Tech. Rep., 1985.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[26] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *arXiv*.

[27] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[28] "Khamlab github page for fed-mec project,," https://github.com/khamfroush-lab/Fed-MEC/tree/master/Distributed-Client-Selection-FL.