Fully decentralized Client Selection for Energy-Efficient Federated Learning

Elia Guerra, Marco Miozzo, and Paolo Dini
Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)
Castelldefels, Spain
{eguerra, marco.miozzo, paolo.dini}@cttc.es

***Abstract*—Federated learning is the most famous algorithm for training machine learning models over distributed datasets without sharing the local data with a central server. However, in many real-life settings, not all clients are equally beneficial for training the model. The selection of the optimal subset of participating clients is critical for ensuring high performance and low energy consumption. This paper formulates the client selection problem as a multi-agent optimization task with the goal of finding a trade-off between performance and energy consumption. In particular, we propose a fully decentralized client selection policy based on non-stationary multi-armed bandits where clients autonomously decide to participate or not in the training process without relying on the central server. The proposed solution outperforms the random client selection policy reducing up to 12% both the number of rounds required to achieve the target accuracy and energy consumption.**

***Index Terms*—federated learning, data analysis, multi-agent systems, multi-armed bandit, energy efficiency, edge computing.**

I. Introduction

Federated Learning (FL) enables training Machine Learning (ML) models over distributed data sources without sharing the local data with a central server. Differently from the standard cloud-based training process it reduces latency, communication overhead, energy consumption, and preserves privacy [1]. FL finds applications in scenarios like healthcare [2], traffic prediction [3], and mobile traffic estimation [4].

The prominent implementation of FL is centralized [5], i.e., a central server coordinates the training process among several clients. At each round, the central server randomly selects a subset of the available clients to train on their local datasets a ML model and share back the updated model weights. Then, the central server is also in charge of aggregating the model updates. Usually, the aggregation method is based on Federated Averaging (FedAvg) [6].

Defining a proper client selection strategy is crucial to guarantee fast convergence, limit communication overhead and reduce energy consumption [7]. In the literature, the client selection problem has been widely investigated (see Section II) focusing on a scenario where the central server has the additional overhead of selecting the optimal subset of clients. Most of the proposed approaches concentrate on maximizing performance, i.e., accuracy or convergence time, without considering that in real-life scenarios energy devices have limited energy available and it should be properly managed to avoid battery depletion and jeopardize model training. Moreover, efficient learning systems, especially during their training phase, is a need today due to the high energy consumption of ML computations [8], [9].

In addition, the classical single point of failure problem may arise in such star topology, where a central node is orchestrating the whole process. It may also represent the bottleneck of the procedure when the number of participating nodes increases. Hence, we focus on relaxing the FL algorithm dependencies on the central server by minimizing its tasks. In particular, in this work, we propose a fully decentralized solution for the client selection problem. Differently from the literature reviewed, we consider an approach where each client autonomously decides to join or not the next FL round. The goal is to find a global trade-off between the achieved performance and the amount of energy spent. The problem is stated as a multi-agent optimization and solved with a model-free approach. In fact, we provide a solution in which each client learns the policy to follow (i.e., to join or not the training process every round) through a lightweight Multi-Armed Bandit (MAB) model. This is the first attempt to apply decentralized decision-making to the client selection problem in FL solutions with a central server, to the best of our knowledge. Simulation results demonstrate that the MAB agents at each client are converging to stable policies and lead to a proper trade-off between the accuracy of the model and the energy spent during the training phase. Our analysis opens new possibilities of improvement specially in presence of the non-iid data (as specified in Section V) and may pave the way for new approaches to decentralized FL.

Summarizing, the contributions of this work may be listed as follows:

- We state the client selection optimization as a multi-agent decision-making problem.
- We propose a fully decentralized client selection policy based on the non-stationary MAB model.

This publication has been partially funded by the Spanish project PID2020-113832RB-C22(ORIGIN)/MCIN/AEI/10.13039/50110001103, European Union Horizon 2020 research and innovation programme under Grant Agreement No. 953775 (GREENEDGE) and the grant CHIST-ERA-20-SICT-004 (SONATA) by PCI2021-122043-2A/AEI/10.13039/501100011033

979-8-3503-0460-2/23/$31.00

- We evaluate the proposed solution and compare it against the random client selection policy.
- We present a set of results, which demonstrate that the proposed solution converges to a stable policy and jointly optimizes the energy consumption and the accuracy of the FL algorithm.

The rest of the paper is organized as follows. Section II describes the related work, Section III describes the FL setting, Section IV introduces the system model and details the proposed solution. Finally, Section V evaluates the algorithm performance, and Section VI concludes the paper with the final remarks.

II. Related work

Client selection for FL is a well-known problem in the literature and it has been addressed from several points of view. A common approach is estimating the client contribution by evaluating specific metrics of its dataset [10], [11]. In [10], the authors define the dataset entropy to capture the distribution, the quantity of information, the unbalanced structure, and the *non-IIDness* of each client dataset. The notion of entropy is used to weigh the local updates of each client during the aggregation phase or to select the subset of participating clients in each FL round. A N-IID Index is introduced in [11] to properly select clients. At each round, the server chooses half of the available clients with the highest number of samples on their local datasets. Then, each selected client sends the mean and standard deviation of each feature to the central server. With this information, the central server can compute the N-IID Index that represents the disparity between two datasets' distributions. The central server uses this index to select the clients with the closest distribution to avoid heterogeneous datasets that may compromise the model aggregation step. A different approach is used in [12] to estimate the contribution of each client to the model convergence. The central server samples a subset of clients with probability proportional to their local dataset size, then the selected clients evaluate the current global model on their local dataset. Finally, the server selects the clients with the highest local losses. An improvement of this algorithm, taking into account communication efficiency and fairness is proposed in [13] with the usage of MAB.

Another common technique adopted to solve the client selection problem is reinforcement learning. In [14], the authors introduce a deep reinforcement learning model running at the central server to find the best client selection policy at every round. An alternative formulation is proposed in [15], where the authors consider also devices with a limited amount of energy units available. The proposed strategy is able to reduce the number of energy units consumed by edge devices. Even in this case, the central server is in charge of selecting participating clients. Both proposals suffer from the curse of dimensionality due to the large action-state space of the centralized control approach and deep neural networks are used as function approximation to limit such issue. This approach may lead to high computational overheads of the central server, which may jeopardize the energy savings of the proposed solution.

Resource-constrained devices are also considered in [16]. Here, client selection is stated as a bilevel optimization problem and resolved with a heuristic approach. In particular, the authors propose a regression-based method to estimate CPU, memory, and energy utilization during the next training round and select devices with enough available resources for training a model to solve an anomaly detection task.

In [7], the authors investigate the impact of temporal patterns on FL performance. Considering a scenario with 10 clients, the authors show that selecting an increasing number of clients for each round guarantees higher accuracy and reduces its standard deviation. Following this observation, the authors propose an online optimization algorithm to jointly optimize the client selection problem and the available communication resources.

From the surveyed literature, it emerges that the client selection problem has been approached mainly with centralized control methods, which, in turns, may lead to the curse of dimensionality problem, especially when the number of clients is high. Moreover, the central server may represent the bottleneck and the single point of failure of the system. To limit such issues, we present a decentralized client selection framework with the aim of improving scalability and flexibility of FL in realistic deployments (e.g., Internet of Things). An energy model based on real measurements is also introduced to drive the training phase towards environmental sustainability. Our approach leverages the client capability of sensing the environment and automatically understanding when to participate in the training round or not, so as to find a global trade-off between the accuracy of the model and the energy spent during the training phase. To the best of our knowledge, this is the first attempt to decentralize the selection of clients in FL solutions with a central server.

III. FL Overview

In FL, a set $\mathcal{N} = \{1, \dots, N\}$ of clients with their local datasets $D_1, \dots, D_N$ train collaboratively a global model, e.g, a set of weights w, to minimize the weighted global loss function:

$$\ell(w) = \sum_{i=1}^{N} \frac{|D_i|}{|D|} \ell_i(w), \tag{1}$$

where ℓ_i is the loss function computed on the local dataset of the $i-th$ client and $|D| = \sum_{i=1}^{N} |D_i|$.

To achieve this goal, clients exchange model parameter updates that have been computed during several training iterations of the global model on the local dataset.

As shown in Figure 1, at the beginning of a round t, the central server selects a random subset $\mathcal{P}^t \subseteq \mathcal{N}$ of m clients (those marked with a green tick in the figure). Each selected client $i \in \mathcal{P}^t$ receives the current global model w^t and trains it on its local dataset for E epochs with a mini-batch size B. At the end of the training process, the trained model w_i^{t+1} is sent to the server. The server aggregates the received local

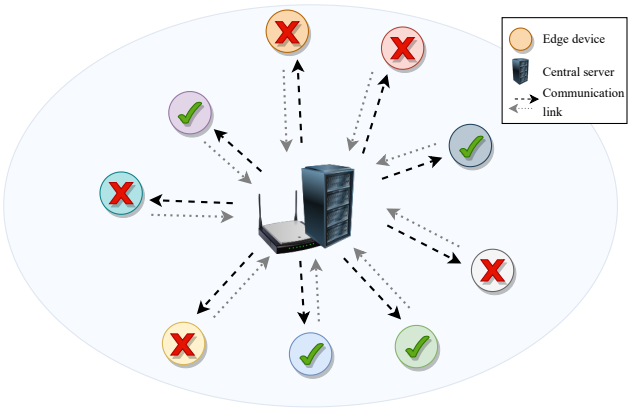

Fig. 1: FL scenario overview.

updates and by computing the weighted average generates the new global model as follows:

$$w^{t+1} = \sum_{i \in \mathcal{P}^t} \frac{|D_i|}{|D|} w_i^{t+1}, \tag{2}$$

where $|D| = \sum_{j \in \mathcal{P}^t} |D_j|$. In this work, we adopt the FedAvg algorithm [6] merging rule to generate global model updates. The process is repeated until the model converges, e.g., a target accuracy is reached.

IV. System Model

In our conception, client selection is seen as a sequential decision-making problem. In the literature it is normally the central server in charge of deciding the active set of clients at every round, usually adopting a random policy. Instead here in this paper, we model the problem as a multi-agent system where every client makes autonomously the decision to participate or not, being the ultimate goal to minimize the energy consumption of the entire process, while at the same time maintaining accuracy at a target level. A model-free cooperative control framework is designed to achieve such goals.

Formally, we consider the FL architecture described in Section III where a set $\mathcal{N} = \{1, \dots, N\}$ of clients with their local datasets $D_1, \dots, D_N$ train collaboratively a machine learning model. Each client is an agent of our sequential decision-making problem. At each learning episode, i.e. FL round, every agent autonomously decides to participate (p) or not (np) in the training process based on its past experience. At the end of each FL round, each client receives a reward from the environment to update its experience. To limit the additional computational overhead on edge devices, we model the agent decision-making as a reinforcement learning task using MAB [17]. In fact, despite its simplicity, the computations needed by MAB result much smaller than other reinforcement learning alternatives (e.g., Q-learning). During a generic round t, a client $i \in \mathcal{N}$, chooses an action $a \in \mathcal{A} = \{p, np\}$ to maximize the long-term reward. To achieve this goal, the client keeps an estimate of the expected reward for each possible action, i.e., $Q_i^t(p)$ and $Q_i^t(np)$. Let $a \in \mathcal{A}$ be the action selected, the client i receives a reward $\mathcal{R}_i^t$ and updates $Q_i^t(a)$ follows:

$$Q_i^{t+1}(a) = Q_i^t(a) + \gamma(\mathcal{R}_i^t - Q_i^t(a)), \tag{3}$$

where γ regulates the trade-offs between the new reward $\mathcal{R}_i^t$ and the old reward estimate $Q_i^t(a)$. Starting from these estimates, the client selects the action for the current round by sampling from the probability distribution obtained with the *softmax policy* [18]:

$$p_i^t(a) = \frac{e^{Q_i^t(a)}}{\sum_{b \in \mathcal{A}} e^{Q_i^t(b)}}, \tag{4}$$

where $p_i^t(a)$ is the probability of selecting the action a at the round t. Note that each client has a different probability distribution.

The reward function for agent i is determined by the difference in validation accuracy of the global model between the current round t and the previous round $t-1$, denoted as $A^t - A^{t-1}$, as well as the total energy spent during the current round $\mathcal{E}^t$, in detail:

$$\mathcal{R}_i^t = (A^t - A^{t-1}) \mathbb{1}_{i \in \mathcal{P}^t} + \left(1 - \frac{\mathcal{E}^t}{\mathcal{E}_{\max}}\right), \tag{5}$$

where $\mathcal{E}_{\max}$ is a normalization factor that represents the maximum energy consumption required if all the N clients would have participated at round t.

The rationale is that of a cooperative multi-agent system, where each client has a global system-wide optimization objective. In our case, the reward function is made of two terms: the first is an accuracy maximization, the second is a penalty due to the energy spent at each round for that accuracy increase. Note that the reward function is different for participating and non-participating clients, as denoted by the presence of the step function $(\mathbb{1}_{i \in \mathcal{P}^t})$ in the accuracy maximization term. This encourages clients to participate and makes the learning process converge faster. Figure 2 shows the process from the perspective of a participating and non-participating client.

The system evolves as follows. At every round t, depending on the decision made, each client may spend energy of three types: computation, communication, and idle energy. The computation energy is the energy spent for training w^t on the local dataset:

$$\mathcal{E}_{\text{train},i}^t = P_{\text{hw},i}^t T_{\text{train},i}^t, \tag{6}$$

where $P_{\text{hw,i}}^t$ is the average power consumed by the hardware during the training process and $T_{\text{train},i}^t$ its duration.

The communication energy is the energy spent for sharing the model updates w_i^t optimized on the local dataset with the central server. As reported in [1], it can be computed as:

$$\mathcal{E}_{\text{tx}} = P_{\text{tx}} T_{\text{tx}}, \tag{7}$$

where P_{tx} and T_{tx} are the transmission power and time, respectively. Note that the communication energy is fixed since

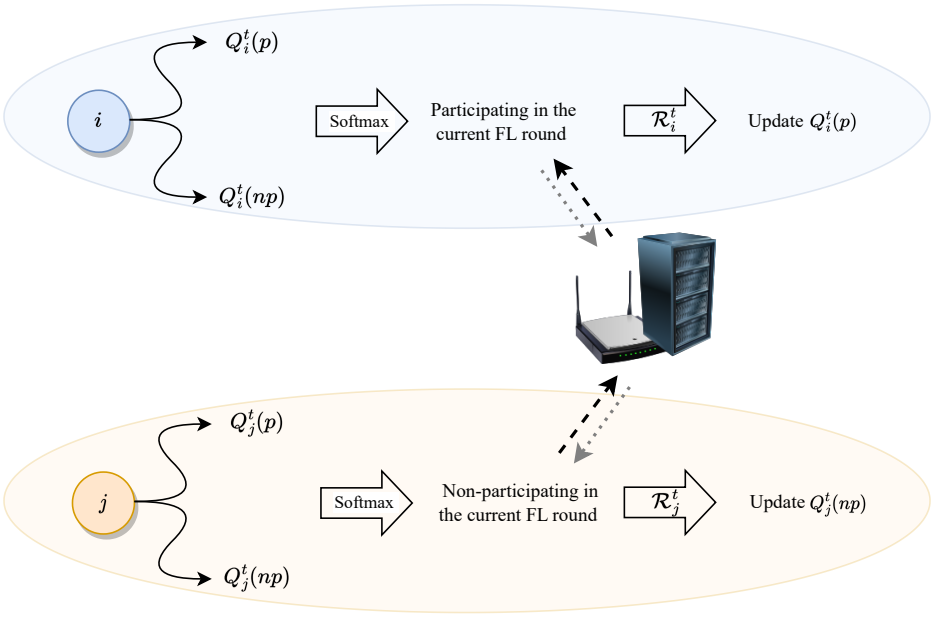

Fig. 2: Client participation over time on CIFAR-10 and CIFAR-10p dataset

all the clients train the same ML model.

The idle energy is the energy spent by the client while waiting for the conclusion of the current FL round:

$$\mathcal{E}_{\text{idle},i}^t = P_{\text{idle},i}(T_{\text{round}} - T_{\text{train},i}^t), \tag{8}$$

where T_{round} is the maximum duration of the round defined by the central server and needed to avoid stragglers: if a client does not start the upload process within T_{round} seconds, its contribution will be discarded. $P_{\text{idle},i}$ is the average power drained while waiting for the conclusion of the current FL round, i.e., $T_{\text{round}} - T_{\text{train},i}^t$.

As a consequence, the total energy for a participating client i during a round t is:

$$\mathcal{E}_{p,i}^t = \mathcal{E}_{\text{train},i}^t + \mathcal{E}_{\text{tx}} + \mathcal{E}_{\text{idle},i}^t. \tag{9}$$

Instead, the energy for a not participating client j is:

$$\mathcal{E}_{np,j} = \mathcal{E}_{\text{idle},j} = P_{\text{idle},j} T_{\text{round}}. \tag{10}$$

The total amount of energy consumed in round t is:

$$\mathcal{E}^t = \sum_{i \in \mathcal{P}^t} \mathcal{E}_{p,i}^t + \sum_{j \in \mathcal{N} \setminus \mathcal{P}^t} \mathcal{E}_{np,j}, \tag{11}$$

where $\mathcal{P}^t$ is the active set of participating clients in round t. The total energy consumed for R rounds is:

$$\mathcal{E} = \sum_{t=0}^{R-1} \mathcal{E}^t. \tag{12}$$

V. Performance Evaluation

A. Simulation Setup

We consider two datasets for our analysis. The first is the CIFAR-10 dataset [19] with 50 000 training samples randomly splitted across 50 clients and 10 000 test samples. From the test set, we extract 7000 samples as a validation set. The input features are 32×32 color images divided into 10 classes. Then, we create also the CIFAR-10p dataset by randomly restricting the client dataset of 46 out of 50 clients to 5 classes only. The clients with full-label distribution are called *superclients*.

TABLE 1: Simulation parameters.

Parameter	Description	Value
$\lvert w \rvert$	Number of model parameters	11 181 642
S_w	ResNet-18 model parameters size	44.73 MB
η	Learning rate	0.01
N	Number of total clients	50
E	Local epochs number	5
B	Batch size	20
T_{round}	Maximum training time	10s
ℓ_i	Local loss function	Sparse Cat. Crossentropy
$T_{\text{acc,IID}}$	Target accuracy on CIFAR-10	0.73
$T_{\text{acc,nIID}}$	Target accuracy on CIFAR-10p	0.66
P_{Tx}	Tx power for edge devices	9 dBm
P_{idle}	Idle power consumption	96.85 W
$\mathcal{E}_{\text{max}}$	Max energy consumption	0.54 Wh

We adopt ResNet-18 [20] to correctly classify the input samples. The number of trainable parameters of ResNet-18 ($\lvert w \rvert$) is 11 181 642 and if stored with *float32* variables, i.e., 4 bytes for each variable, the size in memory (S_w) is 44.73 MB.

At each FL round, each client executes Stochastic Gradient Descent (SGD) for E local epochs and sparse categorical cross-entropy loss (ℓ_i) on the local dataset. Then, it shares the local model updates with the central server. The simulation is concluded when the global model, evaluated on the validation set, archives at least the target accuracy of $T_{\text{acc,IID}} = 0.73$

(a) $Q(p)$ (b) $Q(np)$

Fig. 3: Expected reward on CIFAR-10 dataset.

(a) $Q(p)$ (b) $Q(np)$

Fig. 4: Expected reward on CIFAR-10p dataset.

and $T_{\text{acc,nIID}} = 0.66$ on CIFAR-10 and CIFAR-10p datasets for 3 consecutive rounds, respectively. We require to match the validation accuracy for multiple rounds to avoid validation accuracy spikes. The accuracy thresholds are defined experimentally and consider both the dataset and the ML model.

The list of simulation parameters is reported in Table 1. A more comprehensive overview of the adopted communication model can be found in [1]. Our code, based on PyTorch [21], run on a server equipped with two Intel Xeon 6230, 188GB of RAM, and an RTX 2080Ti[1]. We used Codecarbon [22] Python library to measure $\mathcal{E}_{\text{train}}$, which returns periodically samples of the power consumption by CPU, DRAM, and GPU. We have performed every simulation 5 times for statistical consistency.

B. Result Analysis

We first evaluate the convergence of the proposed MAB implementation to solve the multi-agent sequential decision-making problem. We optimize the value of γ with grid search.

On the CIFAR-10 dataset, the value of $\gamma = 0.7$ is that returning the best performance. Figure 3 shows the evolution of the expected reward of participating $Q(p)$ and non-participating $Q(np)$ for each client during the whole training process. Each row represents one specific client. After a transitory period at the beginning of the process, we can notice that the Q values are stabilized around one value. Moreover, such value is similar for all the clients since data are iid distributed, therefore selecting one client or another does not provide any advantage in terms of accuracy and energy.

Similarly, on CIFAR-10p dataset, the value of $\gamma = 0.6$ is that returning the best performance. Figure 4 shows the evolution of $Q(p)$ and $Q(np)$ for each client during the learning process. As observed for the CIFAR-10 dataset, the values are changing round after round till reaching convergence. So, even in this case, the proposed MAB agent on each client is learning the structure of the problem.

Also, in this case, $Q(p)$ and $Q(np)$ converge to similar

[1]For the sake of reproducibility the source code is available at https://gitlab.cttc.es/supercom/energy-efficient-decentralized-client-selection

TABLE 2: Baseline results on CIFAR-10 and CIFAR-10p dataset with random selection policy.

m	$\overline{R}$	$\sigma(R)$	$\overline{\mathcal{E}}$	$\sigma(\mathcal{E})$
5	62	0	885.21	0.15
10	43	0	649.49	0.18
15	45	0	717.19	0.21
20	49	0	820.84	0.22
25	38	0	667.84	0.24
30	38	0	699.22	0.38
35	35	0	672.66	0.87
40	37	0	741.81	1.46
45	40	0	833.68	1.67
50	39	0	845.44	0.56
MAB	36.2	0.44	638.16	7.14

(a) CIFAR-10

m	$\overline{R}$	$\sigma(R)$	$\overline{\mathcal{E}}$	$\sigma(\mathcal{E})$
5	100	0	1388.81	0.03
10	89	0	1276.24	0.14
15	67	0	990.17	0.41
20	61	0	928.62	0.08
25	55	0	861.72	0.50
30	64	0	1032.50	3.53
35	63	0	1040.60	1.02
40	57	0	967.98	0.89
45	65	0	1130.39	1.38
50	55	0	981.25	0.25
MAB	48.8	2.19	758.79	33.88

(b) CIFAR-10p

values for all the clients. This includes the 4 superclients, i.e. $\{45, 46, 47, 48\}$, which have a local dataset with all the possible labels. When a superclient participates in the process, it leads to a greater accuracy improvement (first addend of (5)). However, its participation also results in higher energy consumption due to its larger dataset (second addend of (5)). In our simulation scenario, these two factors return similar values, thus resulting in equal expected reward for clients with pruned label distributions and superclients. In fact, the reward function is not able to capture the different data distributions across the clients and is missing to let superclients participate more, as should be expected. This opens the doors to further improvements in future work to include additional terms in the reward, which may consider the different label distributions.

As a further analysis, we compare our decentralized approach with the classical centralized random client selection policy. The two key performance indicators used are the number of rounds R and the energy $\mathcal{E}$ required to reach the target accuracy. In Table 2, we report the averaged values of rounds $\overline{R}$ and energy $\overline{\mathcal{E}}$ together with $\sigma(R)$ and $\sigma(\mathcal{E})$ that indicate the relevant standard deviations, respectively. The last row in the Table specifies the values obtained with our MAB agent implementation.

First, we note that on the CIFAR-10 dataset (Table 2a), $m = 35$ minimizes the number of rounds, but the energy consumption is minimized by $m = 10$. This result underlines the importance of having the right client selection policy for achieving a good trade-off between accuracy and energy spent and stresses that research directed to only reduce the convergence time/communication rounds is not enough for efficient FL processes. From our experiments, instead, this claim appears to be valid in non-iid scenarios (Table 2b). Then, MAB reaches the target accuracy on average in 36.2 (48.4) rounds with an average energy consumption of 638.16 Wh (758.79 Wh) on CIFAR-10 (CIFAR-10p). On the CIFAR-10 dataset, the MAB approach requires 1 round more, but saves the 1.7% of energy. On the CIFAR-10p dataset, the proposed MAB approach improves both key performance indicators of the 12% with respect to the best baseline for both energy and number of rounds.

Such better performance may be mainly explained through Figure 5, in which we report the number of participating clients for the two datasets CIFAR-10 (iid) and CIFAR-10p (non-iid).

The active set of clients varies at each round and makes the system find the right trade-off between accuracy and energy, following the defined reward function. Moreover, we can notice that the number of participating clients at each round follows similar trends for the two datasets. This result is due to the fact that our MAB implementation does not consider data distribution as a decision metric. As stated above, such a feature deserves further studies to achieve higher energy-efficient FL processes.

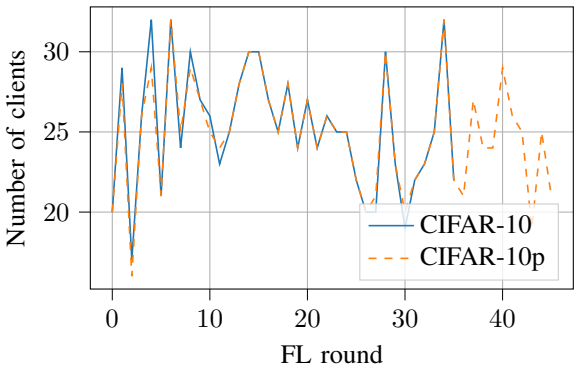

Fig. 5: Client participation over time on CIFAR-10 and CIFAR-10p dataset

VI. Conclusions

This paper addressed the client selection problem in FL by formulating it as a sequential decision-making problem, where each client autonomously decides whether to participate in the current FL round or not. By considering the cost in terms of energy and the potential benefit in the accuracy of each action,

we proposed a novel solution based on non-stationary multi-armed bandits.

The proposed MAB based solution offers several advantages over the previously proposed techniques used for client selection. It enables clients to make autonomous decisions based on their local estimate of the expected reward, promoting a scalable and decentralized client selection policy. This autonomy empowers individual clients to evaluate their own utility in contributing to the global model accuracy. Moreover, the experimental evaluation shows that the MAB-based approach outperforms the baseline by achieving a significant improvement up to up to 12% in energy consumption and number of rounds.

Finally, we are currently working on additional improvements to the definition of the reward function, which aim to consider the heterogeneous data distributions in the local dataset and lead to higher energy efficiency of FL processes.

REFERENCES

[1] E. Guerra, F. Wilhelmi, M. Miozzo, and P. Dini, "The cost of training machine learning models over distributed data sources," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 1111–1126, 2023.

[2] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, pp. 1–19, 2021.

[3] C. Lanza, E. Angelats, M. Miozzo, and P. Dini, "Urban traffic forecasting using federated and continual learning," in *2023 6th Conference on Cloud and Internet of Things (CIoT)*, pp. 1–8, 2023.

[4] V. Perifanis, N. Pavlidis, R.-A. Koutsiamanis, and P. S. Efraimidis, "Federated Learning for 5G Base Station Traffic Forecasting," 2022.

[5] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[7] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1188–1200, 2020.

[8] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3645–3650, 2019.

[9] "AI and Compute," May 2018. https://openai.com/blog/ai-and-compute/.

[10] B. AAMER, H. Chergui, M. Benjillali, and C. Verikoukis, "Entropy-driven stochastic federated learning in non-iid 6g edge-ran," *Frontiers in Communications and Networks*, p. 45, 2021.

[11] A. Houdou, H. Alami, K. Fardousse, I. Berrada, *et al.*, "NIFL: A statistical measures-based method for client selection in federated learning," *IEEE Access*, vol. 10, pp. 124766–124776, 2022.

[12] Y. J. Cho, J. Wang, and G. Joshi, "Client selection in federated learning: Convergence analysis and power-of-choice selection strategies," *arXiv preprint arXiv:2010.01243*, 2020.

[13] 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*, pp. 1066–1069, IEEE, 2020.

[14] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 1698–1707, IEEE, 2020.

[15] H. Zhang, Z. Xie, R. Zarei, T. Wu, and K. Chen, "Adaptive client selection in resource constrained federated learning systems: A deep reinforcement learning approach," *IEEE Access*, vol. 9, pp. 98423–98432, 2021.

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

[17] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[18] R. S. Sutton, A. G. Barto, *et al.*, "Reinforcement learning: an introduction," 1998.

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

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[22] V. Schmidt, K. Goyal, A. Joshi, B. Feld, L. Conell, N. Laskaris, D. Blank, J. Wilson, S. Friedler, and S. Luccioni, "CodeCarbon: Estimate and Track Carbon Emissions from Machine Learning Computing," 2021.