BatteryFL: Battery-Aware Federated Learning

A. Augello, P. Ranjan, A. Gupta, F. Corò, G. Lo Re, S.K. Das.

In Proceedings of GLOBECOM 2025 - 2025 IEEE Global Communications Conference

DOI: https://doi.org/10.1109/GLOBECOM59602.2025.11432132

BatteryFL: Battery-Aware Federated Learning

Andrea Augello[1], Priyesh Ranjan[2], Ashish Gupta[3], Federico Corò[4], Giuseppe Lo Re[1], Sajal K. Das[2]
[1]University of Palermo, Palermo, Italy
[2]Missouri University of Science and Technology, Rolla, USA
[3]BITS Pilani Dubai Campus, Dubai, UAE
[4]University of Padua, Padua, Italy

***Abstract*—Federated learning (FL) has emerged as a transformative paradigm enabling collaborative machine learning without centralizing data, preserving client privacy. This is particularly relevant in the context of edge computing, where the proliferation of Internet of Things devices has led to an explosion of data at the network's edge. These IoT devices, often battery-powered, are limited by their energy capacities, which pose significant challenges for the adoption of FL in such environments. In this paper, we introduce BatteryFL, a novel framework that coordinates battery-aware clients through FL to maximize their contribution to the global model while ensuring a fair distribution of energy consumption across the clients without compromising accuracy. BatteryFL incorporates an innovative data collection algorithm that prioritizes data diversity to minimize battery usage and a sample relevance-based algorithm to select optimal data for training. We also integrate a client selection strategy into the framework to optimize training loss and fairness (based on the battery energy of the clients) simultaneously. Along with a theoretical analysis, we experimentally demonstrate that BatteryFL significantly improves the energy efficiency of FL, prolonging the data collection and the contributions of the clients.**

***Index Terms*—Federated Learning, Energy Efficiency, Client Selection, Performance Fairness.**

I. Introduction

Federated Learning (FL) [1] is a promising paradigm that enables distributed clients to collaboratively train a single model under the coordination of a central server. FL is especially important in edge computing, where the proliferation of IoT devices has shifted data processing to the network edge. FL, despite the several benefits, such as scalability and data privacy, typically assumes that every client has an external power source and a fixed dataset. However, in real-world scenarios, e.g., smart farming (illustrated in Fig. 1), IoT devices such as robots, drones and sensors operate on finite-capacity batteries and are actively involved in data collection [2].

In such dynamic contexts, the energy demands of intensive FL training can rapidly deplete client batteries, leading to dropouts and a decline in model performance [3]. Existing research has explored minimizing training costs [4], yet these approaches typically assume pre-collected datasets, overlooking scenarios with ongoing energy-intensive data collection [2]. The impact of the data collection phase on performance metrics and client battery longevity remains largely unexplored. Thus, comprehensive solutions are needed to co-optimize model accuracy and energy efficiency via client energy management and strategic scheduling, addressing the full data lifecycle.

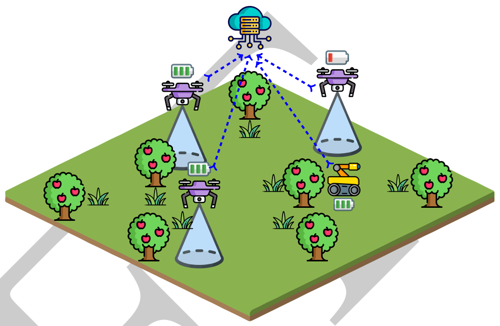


Fig. 1: Example scenario: Battery-constrained edge clients (drones, ground robots) performing active data collection during FL.

Furthermore, client heterogeneity can significantly hamper the performance of FL systems, particularly when not adequately addressed in the client selection process [5]. For example, in highly distributed setups, strategically selecting clients based on a balance between the utility of the dataset (measured by increased training loss) and the training speed has demonstrated significant convergence rate improvements [6]. Moreover, fairness objectives can also be pursued by simultaneously considering factors such as local loss, computational capabilities, and resource demands during client selection [7].

Selecting which data clients use is also a critical yet understudied aspect of efficient FL [8]. Using entire local datasets is inefficient [9], can degrade performance [10], and exacerbates the negative effects of statistical heterogeneity on the global model [11]. Efficient data selection must be lightweight and memory-efficient to suit clients' limited computational resources [12].

The interplay between energy constraints and client heterogeneity introduces further complexity. Clients with lower battery levels might be unable to participate in training for the same duration or with the same amount of data as their more power-rich counterparts [13]. Specific solutions have been proposed to account for this heterogeneity, such as introducing normalization terms when aggregating the models, accounting for the different number of training epochs performed by each client [14]. However, these approaches often do not explicitly consider energy expenditures associated with data acquisition. This paper makes the following major contributions:

- We propose BatteryFL, a novel FL framework that addresses battery constraints by optimizing data collection, sample selection, and fair client selection for energy equity and model quality.

- BatteryFL uses entropy-based data collection,prioritizing diversity and saving energy, and relevance-based sample selection for efficient training energy usage.
- A server-side iterative greedy algorithm selects clients, co-optimizing fairness and training loss.
- BatteryFL, with theoretical guarantees, is validated on benchmarks datasets, showing effective energy management. Its client selection improves FL energy efficiency, extending data collection and client participation.

II. Proposed BatteryFL Framework

This section introduces BatteryFL with formalization of objectives, energy model, and algorithms for client selection.

A. *Objective*

Let $\mathcal{C}$ be the set of all clients. Each client $c \in \mathcal{C}$ has a local dataset D_c^t, containing the samples collected until round t, and a battery level b_c^t. In BatteryFL, only a subset of clients $\mathcal{S}^t \subseteq \mathcal{C}$ participate in any round t. Let $\mathbf{S} = \left[\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^T\right]$ be the sequence of selected clients up to the round $T \geq t$. The client selection process should maximize fairness w.r.t. battery usage between clients while minimizing training loss. At any round t, this process can be expressed as an optimization problem:

$$\mathcal{S}^{t*} = \arg\min_{\mathcal{S}} \left(-f_1(\mathcal{S},t), f_2(\mathcal{S},t)\right), \qquad (1)$$

where,

$$f_1(\mathcal{S},t) = \frac{\left(\sum_{c\in\mathcal{C}\setminus\mathcal{S}}(b_c^{t-1})+\sum_{c\in\mathcal{S}}(b_c^{t})\right)^2}{|\mathcal{C}|\times\left(\sum_{c\in\mathcal{C}\setminus\mathcal{S}}(b_c^{t-1})^2+\sum_{c\in\mathcal{S}}(b_c^{t})^2\right)}, \qquad (2)$$

$$f_2(\mathcal{S},t) = \sum_{c\in\mathcal{C}} \mathcal{L}\left(D_c^t, \left(\frac{1}{|\mathcal{S}^t|}\sum_{c\in\mathcal{S}^t}\theta_c^t\right)\right) - \sum_{c\in\mathcal{C}}\mathcal{L}(D_c^t, \theta^{t-1}). \quad (3)$$

Eq. (2) is the fairness objective derived from Jain's fairness index [15] (JFI): (i) with maximum fairness when all clients have the same battery level ($JFI = 1$), and (ii) minimum fairness when one client has all the battery ($JFI = \frac{1}{|\mathcal{C}|}$). The objective of Eq. (3), with $\mathcal{L}$ the loss, refers to the training loss improvement, defined as the difference between the loss incurred by the global model θ_c^t trained on the selected clients and the loss of the global model θ_{t-1} trained in the previous round. The minimization of f_2 corresponds to obtaining the greatest possible reduction in training loss.

B. *Energy Consumption Model*

We introduce the energy consumption model employed in BatteryFL (partially inspired by [16]). This model enables estimation of the energy usage of each client, facilitating a strategic allocation of the energy budget to various tasks. In each FL round, to participate, clients expend energy to update the local model and exchange model weights with the server.

Let the CPU frequency of client c be q_c, then the power consumed by the CPU of c is $\sigma_c q_c^3$, where σ_c is a coefficient that depends on the switched capacitance of the CPU. Furthermore, given a requirement of N_c^{cpu} CPU cycles to process one sample, the CPU needs to be active for N_c^{cpu} q_c seconds to process the sample. The energy expenditure to train the model for one epoch on a single sample is obtained as the product of power consumption and the time taken to process the sample:

$$E_c^{update} = N_c^{cpu}\sigma_c q_c^2.$$

Next, the energy to upload the model, i.e., energy spent to transmit the model from the client to the server, is given by

$$E_c^{upload} = \frac{p_c s}{B\log_2(1+\frac{p_c h_c^2}{n_0})},$$

where p_c is the transmission power of the client, h_c is the channel coefficient, B is the bandwidth, n_0 is the noise power, and s is the size of the local model, in accordance with [17]. Furthermore, since clients only receive an updated model when they are selected to participate and the downlink bandwidth is usually much higher than the uplink, the energy consumption to download the global model from the server is usually negligible [18], [19] and hence only the upload energy is considered. Client transmission power and CPU frequency are assumed to be heterogeneous across clients but fixed per client.

Algorithm 1 Client Behavior

```
Require: Initial battery $b_c$, diversity threshold $\delta$, number of training samples $n$.
1: $D_c, D_{new} \leftarrow \emptyset, \emptyset$
2: $\theta \leftarrow$ initial model from the server
3: while $b_c > 0$ do
4:   if client is selected to participate then
5:     $\theta \leftarrow$ updated model from the server
6:     if $D_c = \emptyset$ or DIV($D_c$) $< \delta$ or training converged then
7:       while $b_c > |D_{new}| \times E_c^{update} + E_c^{upload}$
             and ((DIV($D_c \cup D_{new}$) $< \delta$
             and $|D_{new}| < n$) or model converged) do
8:         $D_{new} \leftarrow D_{new} \cup$ attempt_sample_collection()
9:         $b_c \leftarrow b_c -$ collection energy
10:    while $b_c > n \times E_c^{update} + E_c^{upload}$
           and training not converged do
11:      $D' \leftarrow$ sample_selection($D_c, D_{new}, n$)
12:      $\theta \leftarrow \theta - \nabla\mathcal{L}(\theta, D')$
13:      Update the reputation of the samples in $D'$
14:      $D_{new} \leftarrow \emptyset$
15:    Upload $\theta$
16:    $b_c \leftarrow b_c - E_c^{upload}$
```

C. *Battery-Aware Clients*

In the BatteryFL framework, each client employs a battery-aware algorithm, outlined in Algorithm 1, that manages its energy consumption during the FL process to prolong lifetime.

The algorithm works as follows: Each client, equipped with an initial battery level (b_c) and an empty local dataset (D_c), participates in the FL process as long as it has sufficient energy. Upon selection, it receives the global model from the server. Then, if the local dataset is insufficient, either because it is lacking in terms of diversity, size, or if the model has already converged for the available samples, the client attempts to collect new samples. This energy-intensive process stops once the diversity/size requirements are met or insufficient energy remains for model training and uploading. After data collection is over, the client selects a subset of samples for training based on their relevance and trains the model on this subset. This selection and training process iterates until the model is convergent or the battery is depleted. At this point, the client uploads the model to the server, and if there is still energy left, the process is repeated. Training is stopped early if the battery approaches depletion, ensuring that the client can still upload the model to the server.

Clients repeatedly attempt data collection, consuming energy for each attempt, then check the remaining battery level and current state of its local dataset to decide whether to continue collecting or start training. In BatteryFL, *data collection stops if the current dataset has a sufficient diversity of samples or if the battery level cannot sustain further collection.*

Let $\mathbf{y}$ be the set of all classes in the dataset, n_y the number of samples of class $y \in \mathbf{y}$ in the dataset, and n the total number of samples to be used for training. The diversity is given by the maximum achievable label entropy of a dataset sampled from the local dataset, which can be expressed as

$$DIV(D_c) = \lim_{x \to \infty} - \sum_{y \in \mathbf{y}} \left(\frac{1}{2} \left(\frac{n}{n_y} \right)^{x} + \frac{|\mathbf{y}|^{x}}{2} \right)^{\frac{-1}{x}} \log \left(\frac{1}{2} \left(\frac{n}{n_y} \right)^{x} + \frac{|\mathbf{y}|^{x}}{2} \right) \tag{4}$$

To train efficiently on growing local datasets while conserving battery, our relevance-based sample selection prioritizes samples by their training loss [9]. This incurs minimal overhead, as the loss is computed during standard training without extra battery cost. The sample selection procedure, detailed in Algorithm 2, works as follows: If the collected samples are new (i.e., never used for training), they should be used for training to assess their relevance. *To achieve a balanced representation of all the classes, some samples from the existing local dataset are also selected if the new samples are not class-balanced and enough battery for the additional samples is available.* We compute the selection probability of any sample i out of n samples using a softmax function with a temperature parameter γ as $Pr(i) = e^{r_i/\gamma} / \sum_{j=1}^{n} e^{r_j/\gamma}$, where, r_i denotes the relevance of i^{th} sample. $\gamma \to \infty$ implies uniform sample selection probability, while $\gamma \to 0$ implies deterministic selection of the most relevant samples per class.

Once the samples are chosen, the client trains the model on this subset for one epoch. During this process, the relevance of each sample is updated using a convex combination of its previous relevance and its importance in the current epoch, governed by an *update factor* α. Unchosen samples receive relevance updates to prevent obsolescence. Specifically, if the relevance of a sample drops below a *threshold* μ, it is updated to a value that blends the threshold with its current relevance, employing a *forgetting factor* β. Clients repeat training on new samples as long as the battery lasts.

D. Server-side Process

While clients prioritize efficient battery use to maximize their contribution, the server focuses on client selection and optimizing fairness and training loss (Eq. (1)). Performing directly the loss objective in Eq. (3) is non-trivial, requiring the server to estimate the loss improvement with a set of clients. An alternative and simpler approach involves maximizing the sum of selected clients' losses, estimated from the previous round's performance, formally expressed by:

$$f_2'(\mathcal{S}, t) = \frac{\sum_{c \in \mathcal{S}} \mathcal{L}\left(D_c^{t-1}, \theta_c^{t-1}\right)}{\sum_{c \in \mathcal{C}} \mathcal{L}\left(D_c^{t-1}, \theta^{t-1}\right)}. \tag{5}$$

The denominator forces $f_2'(\mathcal{S}, t)$ in the $[0, 1]$ interval.

Algorithm 2 Sample Selection and Relevance Update

```
Require: Model $\theta$, dataset $D$, new samples $D_{new}$, update
    factor $\alpha$, forgetting factor $\beta$, temperature $\gamma$, number of
    samples $n$ for training, relevance threshold $\mu$
 1: Initialize all sample relevance to 0, $\{r_i = 0\}_{i=1}^{n}$
 2: for each training round at the client do
 3:     $Pr(i) = e^{r_i/\gamma} / \sum_{j=1}^{n} e^{r_j/\gamma}$
 4:     $n' \leftarrow \max(0, n - |D_{new}|)$
 5:     $\mathbf{y} = \{y | (x, y) \in D \cup D_{new}\}$
 6:     $\mathbf{s} = \emptyset$
 7:     for each class $y \in \mathbf{y}$ do
 8:         $n_{y,new} =$ occurrences of class $y$ in $D_{new}$
 9:         $n_{y,req} = \max(0, \frac{n}{|\mathbf{y}|} - n_{y,new})$
10:         $\mathbf{i}_{(y)} \leftarrow \{i \;\; \text{s.t.} \; (x_i, y) \in D\}$
11:         $\mathbf{s}_y \leftarrow [n_{y,req}$ samples from $\mathbf{i}_{(y)}$ with prob. $\propto Pr(i)]$
12:         $\mathbf{s} \leftarrow \mathbf{s} \cup \mathbf{s}_y$
13:     $D' = [(x_i, y) \in D \;\; \text{s.t.} \; i \in \mathbf{s}] \cup D_{new}$
14:     for each batch $B$ in $D'$ do
15:         Train the model on $B$ and compute the losses
16:         for each $i \in B$ do
17:             $L_i = \mathcal{L}(\theta, x_i, y_i) / \sum_{j \in B} \mathcal{L}(\theta, x_j, y_j)$
18:             $r_i = \alpha r_i + (1 - \alpha) L_i$
19:         Update the model using the loss: $\theta - \nabla \mathcal{L}(\theta, B)$
20:     for each $i \in D \setminus \mathbf{s}$ do
21:         if $r_i < \mu$ then
22:             $r_i = \beta\mu + (1 - \beta) r_i$
23:     $D \leftarrow D \cup D_{new}$
24:     $D_{new} \leftarrow \emptyset$
```

To turn the problem into a maximization problem, we use Eq. (2) as the fairness objective instead of its opposite, Now, the server objective becomes:

$$\mathcal{S}^t = \arg\max_{\mathcal{S}} \left(f_1(\mathcal{S}, t), f_2'(\mathcal{S}, t) \right)$$

Despite its simpler structure, studying this formulation remains challenging due to the exponential number of potential client subsets. An exhaustive search among all $2^{|\mathcal{C}|}$ possible subsets becomes infeasible as the number of clients grows.

To identify the optimal set of clients in a given round, we introduce a greedy algorithm through which the server iteratively adds the client that improves the objectives at each step while removing them from the pool of available clients. Since there are two objectives, instead of a single best client, this procedure yields a non-dominated set of clients. Thus, we use the Chebyshev scalarization method to break ties between clients in the non-dominated set. Given a set of selected clients $\mathcal{S}$, and a set of available clients $\mathcal{S}' = \mathcal{C} \backslash \mathcal{S}$, the greedy objective for Chebyshev scalarization function is:

$$c^* = \underset{c \in \mathcal{S}'}{\arg\max} \left(\min \left(f_1(\mathcal{S} \cup \{c\}, t), f_2'(\mathcal{S} \cup \{c\}, t) \right) \right). \quad (6)$$

Once the scalarization function of Eq. (6) ceases to improve, the algorithm stops and finalizes the client set.

The Chebyshev scalarization ensures that the solutions obtained are at least weakly Pareto optimal [20]. Dominated solutions are removed, yielding the Pareto set. Chebyshev scalarization is employed for final solution selection, offering ($\mathcal{O}(|\mathcal{C}|)$ complexity, but potentially leading to suboptimality compared to methods with $\mathcal{O}(|\mathcal{C}|^2)$) complexity.

Since computing the exact solution is intractable, we provide an analytical approach to estimate the optimality of approximate solutions using a certified dominance bound [21]. We derive the expected value and variance of the fairness and training loss objectives for use in the dominance bound computation under the following **assumptions:** (a) Battery levels and losses are uncorrelated within a single round; (b) Battery levels of distinct clients are uncorrelated; (c) Client battery levels before and after training are distributed as $\sim \mathcal{N}(\mu_{b1}, \sigma_{b1}^2)$ and $\sim \mathcal{N}(\mu_{b2}, \sigma_{b2}^2)$, respectively. Client heterogeneity is compatible with the normal distribution due to the central limit theorem.

When computing the expected value and variance of an objective, we consider a random selection of clients as a set of clients S drawn with equal probability from any of the $2^{|\mathcal{C}|}$ possible subsets of clients. Thus, each client c has a probability of being selected of $\Pr(c) = \frac{1}{2}$.

Theorem 1. *The expected value of the fairness objective obtained by randomly selecting a set of clients can be approximated to*

$$\begin{aligned}\mu_1 =& \Big((|\mathcal{C}|^2(\mu_{b1}^2 + \mu_{b2}^2 + 2)^2 + \\ & 4|\mathcal{C}|(\mu_{b1}^2 + \mu_{b2}^2 + 2))(|\mathcal{C}|(\mu_{b1} + \mu_{b2})^2 + \\ & 2(\sigma_{b1}^2 + \sigma_{b2}^2)\Big)/2|\mathcal{C}|(\mu_{b1}^2 + \mu_{b2}^2 + 2)^3\end{aligned}$$

The variance σ_1^2 of the fairness is upper bounded by:

$$\left(\frac{2}{|\mathcal{C}|\big((\sigma_{b1}^2+\sigma_{b2}^2)+(\mu_{b1}+\mu_{b2})^2\big)} \right)^2 \left(\frac{|\mathcal{C}|}{2}(\mu_{b1}^2 + \mu_{b2}^2 + 2) \right)^{-2} + \frac{\Big(\big(\frac{|\mathcal{C}|}{2}((\sigma_{b1}^2+\sigma_{b2}^2)+(\mu_{b1}+\mu_{b2})^2)\big)^2 |\mathcal{C}|(2+\mu_{b2}^2+\mu_{b1}^2)\Big)}{\big(\frac{|\mathcal{C}|}{2}(\mu_{b1}^2+\mu_{b2}^2+2)\big)^4}.$$

Remark. *As Theorem 1 overestimates the variance, its actual value is lower than the bound, meaning that we obtain a conservative estimate of the certified dominance bound.*

Theorem 2. *The expected value of $f_2'(S)$ is $\mu_2 = 0.5$, with a variance σ_2 of $\frac{1}{8} + \frac{\sigma_l^4 + \mu_l^2}{4\mu_l^2|\mathcal{C}|}$.*

Proofs for Theorems 1 and 2 are omitted due to limited space. These analytical bounds can be used by the server to assess the quality of the solution and, if necessary, to refine it.

III. Experimental Evaluation

To evaluate BatteryFL's efficiency, we conducted simulations on standard ML tasks (MNIST dataset [22]) and a real-world agricultural application, the PlantVillage dataset[1], referred to as Plant. The PlantVillage Dataset contains $50,000$ color images of plant leaves across 38 classes, resized to 128×128. We simulate non-iid data distribution labels between clients with a Dirichlet distribution, and each client, with a predetermined initial battery level, participates in collaborative training until battery depletion. We also report the following metrics: (i) *Loss probability*: probability that the selected clients achieve a better loss objective than a random selection of clients. (ii) *Battery probability*: The probability that the selected clients achieve a better fairness objective than a random selection of clients. (iii) *Dominated probability*: The lower bound probability that the client selection is not dominated by any other solution.

a) Experimental Setup: We simulate BatteryFL using the PyTorch library, experiments on MNIST and Plant datasets used SGD optimizers (learning rates 0.01 and 0.001, respectively). MNIST used a CNN with two convolutional layers and three dense layers, while Plant used the VGG-16 model [23]. We used FedNova [14] for aggregation, Table I summarizes the optimal hyperparameters for the experiments. Communication and single-sample training energy estimates (Section II-B) are detailed in Table I. We tested multiple data collection costs to simulate various scenarios (sensitivity analysis in section III-C). If not specified otherwise, the collection cost corresponds to training on 20 samples.

TABLE I: Experimental parameters

Hyperparameter	Value	Parameter	Value
Update factor (α)	0.25	CPU frequency(f_i)	$U(1.5, 2)$ GHz
Forgetting factor (β)	0.25	Switched Capacitance(σ)	10^{-28} F
Smoothing factor (γ)	1	CPU Cycles(C_i)	$U[5, 15] \times 10^4$
Diversity threshold (δ)	0.7	Bandwidth(B)	5×10^6 Hz
Minimum reputation (μ)	0.5	Transmission Power(p_i)	10 dBm/MHz
Training batch size	64	Channel Gain(h_i)	-60 dB
Training size	500	Noise Power(N_0)	-104 dBm/10MHz

b) Baselines: We experimentally validate our proposed client selection algorithm against **EAFL** [3], an energy-aware framework aiming to optimize accuracy for FL clients with energy constraints. Furthermore, we benchmark against an **All** strategy that selects all clients with the remaining battery for training. All strategies tested utilize the same client behavior as in Algorithm 1. We further compare performance against two genetic algorithm-based approaches as alternatives to our approach: **(i) NSGA-II** [24]: A widely used multi-objective optimization method employing Pareto dominance and a solution population to iteratively refine a set of non-dominated solutions. **(ii) AGE-II** [25]: An evolutionary algorithm for

[1]https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset

multi-objective optimization, AGE-II utilizes approximation-guided evolution to estimate the Pareto front's geometry and guide the search toward non-dominated solutions.

A. *Impact of Battery-aware Client*

As client behavior directly impacts client selection strategies, we first assess the importance of the client-side BatteryFL components: Algorithm 1 and 2. Table II presents the performance of the system with and without these algorithms. Omitting both algorithms simulates a standard FL setting in which clients collect n samples, then train for 10 epochs per round until battery depletion. Removing only Algorithm 1 leads clients to attempting a new sample collection each round, even without model convergence on the current data. These naive approaches display significantly reduced system lifetimes and low model accuracy (often predicting a single class). Hence, for a fair comparison, these algorithms remain active in all subsequent experiments, regardless of the client selection strategy.

TABLE II: Comparison of BatteryFL framework with naive client strategies.

Dataset		BatteryFL	w/o Algorithm 1	w/o Algorithm 1,2
Plant	Test Acc (%)	**90.90**	6.08	6.08
	Setup Lifetime	**38**	27	30
MNIST	Test Acc (%)	**93.63**	9.82	9.82
	Setup Lifetime	**40**	27	26

B. *Trade-off of Objectives*

First, we analyze the client selection process and the corresponding dominance probability for the MNIST dataset; the results obtained are shown in Fig. 2.

Initially, most clients are selected, prioritizing the loss objective over the fairness objective. After the 10th round, the focus starts shifting to the battery objective to extend the setup lifetime, while maintaining a high loss objective. As the setup lifetime reaches its end, the selection process is more stringent, with fewer clients participating in each round. In rounds 20–30, at most two clients per round participate, always ensuring the highest possible fairness. Finally, as clients exhaust their energy, the optimization focuses solely on the loss objective until no client can participate. Analogous results are obtained for the Plant dataset.

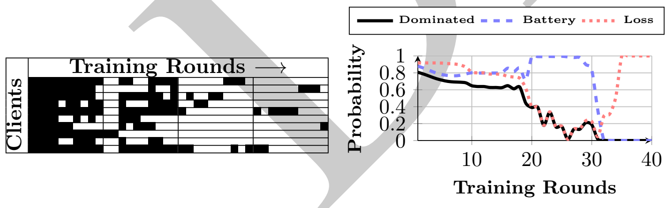
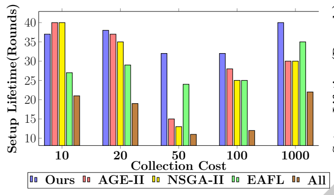


Fig. 2: (Left) Selected clients (black) per round. (Right) Corresponding selection dominance probability. Selection strategy shifts from prioritizing loss to battery objective to extend system lifetime.

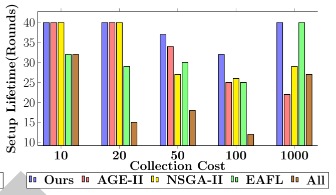
C. *Analyzing Data Collection Cost*

We analyze the effect of data collection cost on system performance, focusing on global model accuracy, data volume, and the number of achievable training rounds.

A single collection operation includes movement, sensor activation, and sample analysis costs. Unlike training and communication battery requirements, the collection cost is not predefined due to its variability between application scenarios. Instead, we define ϱ as the ratio of the cost of collecting a single sample and the cost of training on a single sample. To simulate various scenarios, we vary ϱ in $\{10, 20, 50, 100, 1000\}$.



(a) Plant



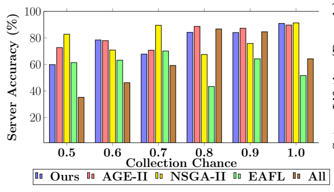
(b) MNIST

Fig. 3: Setup Lifetime results for varied collection cost.

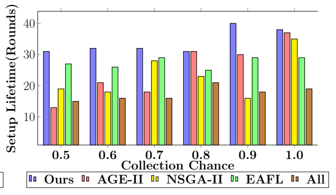
An interesting pattern can be seen in Fig. 3, where the setup lifetime has a non-linear relationship with the collection cost. Initially, the setup lifetime decreases with ϱ, as more and more energy is spent on data collection, leaving less overall available energy. However, for higher collection costs, the setup lifetime increases, as clients cannot collect many samples, reducing training and leading to a longer lifetime. The trend is consistent across all strategies, with our proposed strategy achieving the longest setup lifetime. NSGA-II and AGE-II show less consistent results, sometimes approaching the performance of training with all clients, emphasizing the effectiveness of BatteryFL. Note that NSGA-II and AGE-II differ solely in their server-side implementation, using the same client behavior as Algorithm 1.

D. *Data Collection Probability*

Energy-aware clients have a probabilistic chance of successfully collecting data for each attempt. To gauge the impact of this probability on BatteryFL, we varied it in the $[0.5, 1.0]$ range with 0.1 increments. The uncertainty in the data collection process influences client selection and overall system performance. Fig. 4 shows the results for the Plant dataset. Is is worth noting that reduced collection chances decrease the global model's performance. Indeed, across all scenarios and datasets, our proposed strategy consistently achieves the longest setup lifetime and high model performance.



(a) Server Accuracy



(b) Setup Lifetime

Fig. 4: Results for varied collection chance on the Plant dataset.

E. *Data Diversity*

The diversity of the collected dataset defined in Eq. (4) is a key factor in the energy consumption of the clients. Thus, we analyze the impact of data diversity on the performance of the BatteryFL framework by varying the entropy threshold at which data collection stops. The Dirichlet distribution parameter governs the data distribution among clients, with higher values leading to greater diversity. We vary the diversity threshold in the range $\{0.2, 0.4, 0.6, 0.8\}$ and the parameter (α) in the Dirichlet distribution in the range $\{0.05, 0.1, 0.5, 1, 10\}$. Fig. 5 shows the results of these experiments.

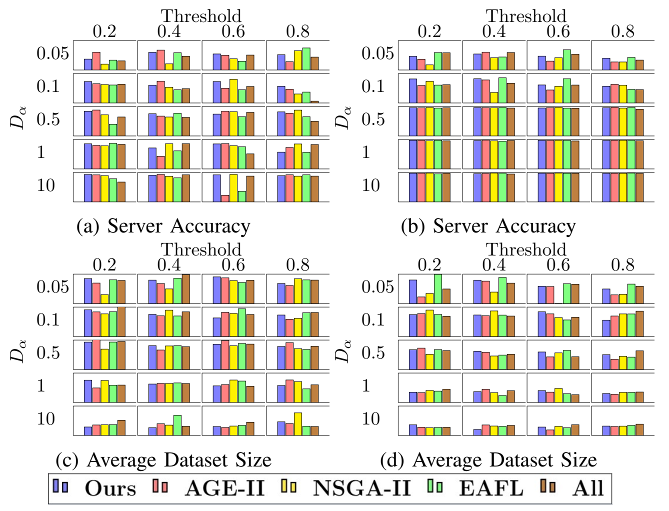


Fig. 5: Results for varied non-IID levels (higher values of parameter D_α indicate greater data diversity among clients) and entropy thresholds for Plant (a,c) and MNIST (b,d).

The results show a modest impact of the diversity threshold on the performance of the BatteryFL framework, suggesting that the dominant component of the data collection decision is the convergence criterion. The label diversity of the clients has a significant impact on the performance of the system, with higher diversity leading to better performance. The increased diversity allows clients to collect less data to meet the entropy target, helping them invest more resources in training.

IV. Conclusion

Federated Learning offers a promising solution for training machine learning models on edge devices, eliminating the need to transmit data to a central server. However, optimizing the energy consumption of these edge devices is crucial. We introduce BatteryFL, an energy-efficient FL framework for battery-powered devices. It enables client-side data optimization and uses a server-side selection algorithm balancing loss and energy fairness. Evaluations confirm high model performance, energy fairness, and system longevity. BatteryFL facilitates sustainable FL under energy constraints. Future work will explore higher device heterogeneity and serverless frameworks where the aggregator can be chosen from the participating clients.

Acknowledgements

The work is supported by BITS Pilani Dubai Campus (BPDC) for the project "FaR-FedIoT: Fair and Resource-adaptive Federated System for IoT" under NFSG scheme.

References

[1] 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, pp. 1273–1282.

[2] F. B. Sorbelli, F. Corò, S. K. Das, L. Palazzetti, and C. M. Pinotti, "Drone-based bug detection in orchards with nets: A novel orienteering approach," *ACM Trans. on Sensor Networks*, 2024.

[3] A. Arouj and A. M. Abdelmoniem, "Towards energy-aware federated learning on battery-powered clients," in *ACM wkshp on Data Privacy and Federated Learning Technologies for Mobile Edge Network*, 2022.

[4] R. Chen, L. Li, K. Xue, C. Zhang, M. Pan, and Y. Fang, "Energy efficient federated learning over heterogeneous mobile devices via joint design of weight quantization and wireless transmission," *IEEE Trans. on Mobile Computing*, 2022.

[5] C. Yang, M. Xu, Q. Wang, Z. Chen, K. Huang, Y. Ma, K. Bian, G. Huang, Y. Liu, X. Jin *et al.*, "Flash: Heterogeneity-aware federated learning at scale," *IEEE Trans. on Mobile Comp.*, 2022.

[6] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *Symposium on Operating Systems Design and Implementation*, 2021, pp. 19–35.

[7] A. Sultana, M. M. Haque, L. Chen, F. Xu, and X. Yuan, "Eiffel: Efficient and fair scheduling in adaptive federated learning," *IEEE Trans. on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4282–4294, 2022.

[8] A. Li, L. Zhang, J. Tan, Y. Qin, J. Wang, and X.-Y. Li, "Sample-level data selection for federated learning," in *IEEE Conf. on Computer Communications*, 2021, pp. 1–10.

[9] J. Shin, Y. Li, Y. Liu, and S.-J. Lee, "Fedbalancer: Data and pace control for efficient federated learning on heterogeneous clients," in *Int. Conf. on Mobile Systems, Applications and Services*, 2022, pp. 436–449.

[10] C. Gong, Z. Zheng, F. Wu, Y. Shao, B. Li, and G. Chen, "To store or not? online data selection for federated learning with limited storage," in *ACM Web Conf. 2023*, 2023, pp. 3044–3055.

[11] A. Augello, G. Falzone, and G. Lo Re, "Dcfl: Dynamic clustered federated learning under differential privacy settings," in *Pervasive Computing and Communications wkshps*, 2023, pp. 614–619.

[12] C. Gong, Z. Zheng, Y. Shao, B. Li, F. Wu, and G. Chen, "Ode: An online data selection framework for federated learning with limited storage," *IEEE/ACM Trans. on Networking*, 2024.

[13] K. Pfeiffer, M. Rapp, R. Khalili, and J. Henkel, "Federated learning for computationally constrained heterogeneous devices: A survey," *ACM Computing Surveys*, vol. 55, no. 14s, pp. 1–27, 2023.

[14] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[15] R. K. Jain, D.-M. W. Chiu, W. R. Hawe *et al.*, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, vol. 21, p. 1, 1984.

[16] R. Albelaihi, L. Yu, W. D. Craft, X. Sun, C. Wang, and R. Gazda, "Green federated learning via energy-aware client selection," in *IEEE Global Communications Conf.*, 2022, pp. 13–18.

[17] N. H. Tran, W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *computer communications conf.* IEEE, 2019, pp. 1387–1395.

[18] J. Feng, L. Liu, Q. Pei, and K. Li, "Min-max cost optimization for efficient hierarchical federated learning in wireless edge networks," *IEEE Trans. on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2687–2700, 2021.

[19] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning in mobile edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3606–3621, 2021.

[20] I. Das and J. E. Dennis, "A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems," *Structural optimization*, pp. 63–69, 1997.

[21] D. Berend, S. S. Skiena, and Y. Twitto, "Dominance certificates for combinatorial optimization problems," *Optimization Problems in Graph Theory*, pp. 107–122, 2018.

[22] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural comp.*, vol. 1, no. 4, pp. 541–551, 1989.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[24] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in *Parallel Problem Solving from Nature*, 2000, pp. 849–858.

[25] M. Wagner and F. Neumann, "A fast approximation-guided evolutionary multi-objective algorithm," in *Genetic and evolutionary computation*, 2013, pp. 687–694.