POLITECNICO DI TORINO
Repository ISTITUZIONALE

FedAGF: Adaptive Concurrency via Gradient Feedback for Mitigating Extreme Label Skew in Budget-Constrained Federated Learning

Original
FedAGF: Adaptive Concurrency via Gradient Feedback for Mitigating Extreme Label Skew in Budget-Constrained Federated Learning / Malan, E., Peluso, V., Calimera, A., Macii, E.. - In: IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS. I, REGULAR PAPERS. - ISSN 1549-8328. - ELETTRONICO. - 73:6(2026), pp. 3835-3848. [10.1109/tcsi.2026.3663070]

Availability:
This version is available at: 11583/3007627 since: 2026-02-15T15:30:56Z

Publisher:
IEEE

Published
DOI:10.1109/tcsi.2026.3663070

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

26 November 2026

FedAGF: Adaptive Concurrency via Gradient Feedback for Mitigating Extreme Label Skew in Budget-Constrained Federated Learning

Erich Malan, *Graduate Student Member, IEEE*, Valentino Peluso, *Member, IEEE*, Andrea Calimera, *Member, IEEE*, Enrico Macii, *Fellow, IEEE*

***Abstract*—Cross-device Federated Learning (FL) enables large fleets of distributed edge devices to collaboratively train a global classification model without sharing their data. The resulting training quality is strongly limited by *extreme label skew*, a condition where each device holds samples from a subset of the target classes. In such cases, local updates become biased toward local distributions, slowing down convergence and degrading the global model accuracy. These effects get critical when devices operate under constrained energy budgets that restrict their participation to a limited number of synchronization rounds, further reducing the achievable accuracy. To overcome these limitations, we introduce Federated Learning with Adaptive Concurrency via Gradient Feedback (FedAGF), a control policy that dynamically adjusts the number of devices selected for synchronization throughout training. FedAGF adapts to training dynamics by monitoring global model updates and elevating participation whenever progress slows. This adaptive mechanism balances accuracy improvement with efficient use of devices' energy budgets, allocating resources when they provide the greatest benefit to convergence. Extensive experiments on CIFAR-10 and CIFAR-100 demonstrate that FedAGF effectively mitigates extreme label skew, achieving up to 14.14% higher accuracy than state-of-the-art FL methods and enabling efficient and scalable training, even under skewed label distributions and resource constraints.**

***Index Terms*—Federated learning, energy efficiency, extreme label skew, adaptive concurrency, budgeted training.**

I. Introduction

CROSS-device Federated Learning (FL) is a distributed training paradigm where large populations of mobile and IoT devices collaboratively learn a global model without exposing their private data, thus complying with data protection regulations [1]. A central server coordinates the process through a sequence of synchronization rounds, each organized in three main stages. First, the server selects a subset of available devices to participate in the synchronization round. Then, the selected devices download the current global model, perform local training on their private datasets, and return their model updates to the server. Finally, the server aggregates the received updates to produce a new global model version before proceeding to the next round.

FL has demonstrated effectiveness in many IoT applications, like image classification [2], hand gesture recognition [3], keyword spotting [4], facial expression recognition [5], smart meter data profiling [6], and scheduling of distributed energy resources [7]. However, its application in cross-device scenarios is still challenged by three tightly coupled issues: *extreme label skew*, *partial participation*, and *constrained energy budgets*. Extreme label skew arises when the data available on each device covers only a fraction of the target classes, a quite common case in real scenarios. For example, in intrusion detection systems based on network logs, a device typically records only a limited set of traffic events, and no device observes all possible attack types [8]. Under such conditions, local models tend to overfit to the observed classes, producing biased updates that slow convergence and reduce the accuracy of the global model. The problem is exacerbated by the partial participation scheme that characterizes cross-device FL: in each round, the server can select only a subset of devices for synchronization due to communication and scalability limits [9]. The number of participating devices per round is referred to as *concurrency*, which in standard FL systems is chosen as a fixed hyperparameter for the entire training process. Low concurrency limits the diversity of training labels per round, and, under skewed distributions, certain classes may remain under-represented across multiple rounds, further degrading model accuracy. Increasing concurrency can mitigate this effect by raising the class diversity represented in each round, but mobile and IoT devices are typically battery-powered and operate under constrained energy budgets [10]. Maintaining high concurrency might set excessive resource demand, reducing the total number of possible rounds [11]. As a result, the training often ends with an under-optimized global model, as the available device budgets are insufficient for the number of rounds required to reach higher accuracy.

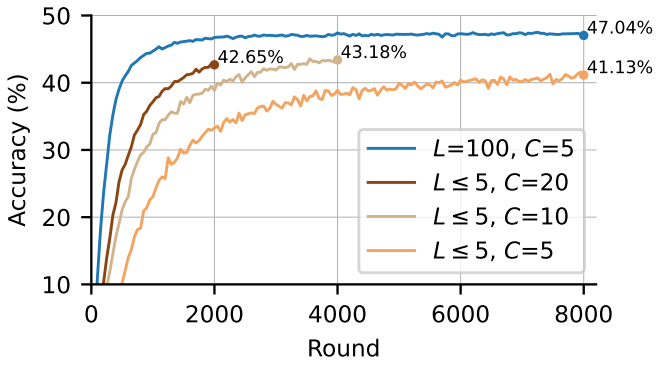


Fig. 1. Accuracy during training on CIFAR-100 with different label distributions (L) and concurrency levels (C) [12].

A quantitative analysis is reported in Fig. 1. The plot

Erich Malan, Valentino Peluso, and Andrea Calimera, are with the Department of Control and Computer Engineering, Politecnico di Torino, 10129 Turin, Italy (e-mail: erich.malan@polito.it; valentino.peluso@polito.it; andrea.calimera@polito.it).

Enrico Macii is with the Interuniversity Department of Regional and Urban Studies and Planning, Politecnico di Torino, 10129 Turin, Italy (e-mail: enrico.macii@polito.it).

shows the accuracy curves of a five-layer convolutional neural network trained on CIFAR-100 across 100 devices using FedAvg [9] with partial participation of C clients per round. We considered two label distribution scenarios. The all-class case ($L = 100$), where each device holds samples from the entire set of target labels, and the skewed case ($L \leq 5$), where each device holds samples from a subset of at most L target labels. In both cases, we assumed each device can participate in up to 400 rounds due to energy constraints. In the all-class case, the global model accuracy improves quickly, even with a relatively low concurrency C of five clients per round ($C = 5$). In contrast, under extreme label skew, the accuracy drops significantly. For instance, with $L \leq 5$ and $C = 5$, convergence is considerably slower, and the final accuracy is significantly lower than in the all-class case (41.13% vs. 47.04%). To further investigate the interplay between label skew and client participation, we repeated the experiment under different concurrency settings, exploring $C \in \{5, 10, 20\}$. Higher concurrency partially alleviates the effect of label skew, as more classes are represented within each round, leading to faster convergence during the early training phase. However, increasing concurrency also increases the budget consumption rate, since devices are required to participate more frequently. Consequently, the total number of rounds is reduced (e.g., 4000 rounds at $C = 10$, 2000 at $C = 20$), preventing further model refinement. Indeed, while $C = 10$ achieves higher accuracy than $C = 5$ (43.18% vs. 41.13%), further increasing concurrency to $C = 20$ results in lower final accuracy due to budget shortage (42.65%). These findings reveal the intrinsic limitation of a static concurrency selection, which is the standard approach in current FL systems. A fixed concurrency forces a suboptimal tradeoff between accuracy and resource usage: low concurrency allows for more rounds but suffers from class under-representation, while high concurrency mitigates skew-induced bias but quickly drains clients' resources. Neither option provides an efficient path to accurate models under budget constraints, motivating the need for adaptive strategies that dynamically control concurrency.

Accordingly, the research gap we address is twofold. First, existing FL approaches lack mechanisms to *sense* when additional client participation is necessary to sustain learning progress. Second, they do not implement *energy-aware control policies* that allocate more resources when this leads to a benefit to convergence.

We bridge this gap by proposing Federated Learning with Adaptive Concurrency via Gradient Feedback (FedAGF), a protocol that dynamically regulates client participation according to the training dynamics of the global model. The key intuition is that in the early rounds, when the model is far from convergence, low concurrency already provides effective updates while conserving client resources. As training proceeds, however, local updates may no longer be sufficient to bring improvement, particularly under extreme label skew. Here is where a larger concurrency increases label representation in each round, reducing bias and accelerating convergence. To enable adaptive concurrency, FedAGF implements a lightweight monitoring mechanism that tracks the evolution of the global model's pseudo-gradient, combined with a heuristic control policy to detect stagnation phases when progress stalls and promptly increase the number of participating devices. FedAGF reaches a better balance between accuracy improvement and resource usage, ensuring client budgets are allocated to rounds where they have the greatest impact on the global model quality.

The main advances introduced by our work are summarized as follows:

- We identify and formalize the *adaptive concurrency* problem in cross-device FL under extreme label skew and constrained energy budgets.
- We propose FedAGF, which introduces a gradient-based feedback signal to detect stagnation and dynamically regulate client participation.
- We demonstrate that, by reacting to the evolving state of the global model, FedAGF automatically adapts the concurrency schedule to diverse data and system conditions, including different dataset complexity, degrees of label skew, numbers of local training iterations, and client populations, achieving higher accuracy while meeting the budget constraints.

We validate the effectiveness of FedAGF through extensive experiments on image classification tasks using the CIFAR-10 and CIFAR-100 datasets. The evaluation assesses different degrees of label skew, multiple concurrency settings, and varying budget constraints to reflect the typical conditions of cross-device deployments. The collected results show that FedAGF consistently outperforms state-of-the-art baselines [9], [13]–[16], delivering higher accuracy with more efficient resource utilization. Notably, FedAGF achieves accuracy gains of up to 14.14% over existing methods, demonstrating its potential as an efficient and scalable solution for cross-device FL.

To notice that this work extends our previous studies [12] by providing a more detailed presentation of the FedAGF protocol together with an expanded experimental evaluation under a wider range of conditions. Specifically, the new contributions introduced in this work are:

- A more comprehensive review of existing mitigation strategies for extreme label skew in FL (Section II), enabling a clearer positioning of FedAGF within the state-of-the-art and highlighting the novelty and advantages of our approach.
- A thoroughly revised and more complete presentation of the FedAGF methodology (Section III), including: (i) preliminaries on budget-constrained FL (Section III-A); (ii) the energy model and the associated optimization problem (Section III-B); (iii) a more detailed derivation of the FedAGF protocol (Section III-C); (iv) extended implementation details supported by new pseudo-codes (Algorithms 2, 3, and 4) and a workflow overview (Sections III-D and III-E); (v) a convergence analysis of FedAGF (Section III-F).
- A substantially expanded experimental evaluation (Section V). In addition to the core results presented in [12], this work includes: (i) an enhanced sensitivity analysis of FedAGF hyperparameters (Section V-B); (ii) robustness and scalability studies under diverse system configu-

rations (Section V-C); (iii) an evaluation of FedAGF under gradient compression via low-precision training (Section V-D); and (iv) validation under heterogeneous energy budgets across devices (Section V-E).

The remainder of the paper is organized as follows. Section II reviews related work on energy-efficient FL and existing approaches for mitigating extreme label skew, including the methods used for comparison. Section III revisits the standard FL workflow and presents the design of FedAGF, detailing its monitoring mechanism and control policy. Section IV describes the experimental setup adopted for validation. Section V reports the results and discusses the main findings. Section VI concludes by summarizing the key contributions.

II. Related Work

A. Energy-Efficient FL

Training deep learning models in cross-device FL typically requires thousands of synchronization rounds before convergence, which translates into high energy costs for participating clients. To address this issue, a substantial body of research has focused on improving the energy efficiency of FL by reducing the computational and communication overhead of the training process. A common approach is model compression, which reduces the volume of exchanged data for synchronization and simplifies local training. Representative techniques include gradient sparsification [17]–[19], low-precision training [20]–[23], and partial synchronization strategies that freeze selected layers [24]–[26] or parameters [27]. Although effective in reducing energy consumption, these methods often incur an accuracy penalty, especially under extreme label skew [21]. As such, model compression alone is insufficient to guarantee robust training in realistic cross-device deployments, where limited client budgets and skewed data distributions coexist.

B. Mitigating Extreme Label Skew

Several works have directly addressed extreme label skew, aiming to reduce its negative impact on convergence and accuracy. These approaches can be grouped into four main categories: (i) *local optimization*, where the on-device training process is adjusted to produce less biased client updates; (ii) *global aggregation*, which employs alternative model aggregation schemes more robust to heterogeneous label distributions; (iii) *data augmentation*, where external or synthetic training samples are added to local datasets to increase class coverage; and (iv) *synchronization mode*, which includes sequential training and coordination protocols.

1) Local Optimization: Optimizing local training is among the most studied strategies to mitigate extreme label skew. Standard cross-entropy loss, widely used in classification tasks, tends to amplify bias toward the classes present in each client, producing weak model updates. Several techniques modify the local loss function to reduce bias and accelerate convergence [13], [14], [28]–[31]. For example, FedRS [13] introduces rescaling factors in the softmax operator of the cross-entropy loss to suppress gradients associated with missing classes, while FedLC [14] recalibrates local predictions for a better alignment with the global model. Other methods leverage knowledge distillation from the global model to transfer information about absent classes to local models [32], [33], yet at the cost of additional energy due to the extra forward pass required in each training iteration. FedVLS [15] combines both strategies, applying a reweighted loss based on local class frequencies together with distillation on missing classes. Complementary to these techniques, FedBSS [34] proposes an heuristic sample selection strategy that restricts local training to dynamically chosen subsets of data, filtering out samples deemed to exacerbate model drift. However, this heuristic introduces extra computation on clients, limiting its applicability in resource-constrained environments.

2) Global Aggregation: Another direction is to modify the aggregation process at the server side. A representative example is FedConcat [35], which clusters clients with similar label distributions and trains separate global models for each cluster. These models are then merged into a unified one at the end of training. The key limitation of FedConcat is that it relies on full client participation to form representative clusters, which is impractical in cross-device FL. Moreover, the final model complexity grows with the number of clusters, raising concerns about deployment on resource-constrained devices. Recently, FedCDA [36] proposes the aggregation of local models collected across multiple rounds, aiming to preserve the knowledge from clients selected in the earlier rounds.

3) Data Augmentation: Augmenting local datasets with additional samples from missing classes is another option. This can be achieved by retrieving data from public repositories [37] or by generating synthetic examples [38], [39]. While effective in improving class coverage, these methods depend on the availability of representative external data and require extra energy for data retrieval or generation, limiting adoption in cross-device scenarios.

4) Synchronization Mode: Synchronization typically occurs with parallel updates, where selected clients train and upload their models concurrently. As an alternative, sequential training allows clients to update the global model one after another within the same round, passing the intermediate model from one client to the next before the final version is uploaded to the server. Recent works explore sequential training as an answer to skewed label distributions. One representative case is FedSE [40], which forms clusters of clients such that each cluster covers all target classes. Within each round, clients in a cluster iteratively train a shared model before uploading it to the server for global aggregation, thereby ensuring that each cluster model is trained over the full label space. SAFA [41] further extends this idea by introducing dynamic interactions among clients, replacing fixed clusters with adaptive coordination. However, despite their potential to reduce model bias, the authors of both FedSE and SAFA explicitly acknowledge that sequential training incurs substantial communication and energy overhead due to frequent inter-client synchronization, which makes these approaches less suitable for resource-bounded cross-device FL.

Algorithm 1: FedAvg with energy-constrained clients.

1 $\mathbf{w}_1 \leftarrow$ Random values
2 $t \leftarrow 1$
3 **while** *Devices have remaining energy* **do**
4 Randomly select a subset $\mathcal{S}_t \subset \mathcal{S}$ of C clients from devices with available energy
/* Training & Synchronization */
5 **for** $s \in \mathcal{S}_t$ **do in parallel**
6 DOWNLOAD($\mathbf{w}_t$)
7 $\mathbf{w}_{s,t} \leftarrow$ TRAIN($\mathbf{w}_t$, $\mathcal{D}_s$, K)
8 $\mathbf{\Delta}_{s,t} \leftarrow \mathbf{w}_{s,t} - \mathbf{w}_t$
9 UPLOAD($\mathbf{\Delta}_{s,t}$)
/* Global Aggregation */
10 $\mathbf{\Delta}_t \leftarrow \frac{1}{|\mathcal{S}_t|} \sum_{s \in S_t} \mathbf{\Delta}_{s,t}$
11 $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \mathbf{\Delta}_t$
12 $t \leftarrow t + 1$

C. Adaptive Concurrency

In a standard FL scheme, the number of participating clients during a round is typically kept fixed. Only a few works investigate dynamic adjustments of concurrency to mitigate skewed distributions. AdaFL [16] represents an early attempt in this direction. It implements a stepwise strategy by which concurrency increases at regular intervals. Although easy to use, AdaFL ignores the actual learning dynamics of the global model and fails to adapt to variations in convergence speed or dataset complexity. FedDR [42] proposes a budget-driven policy in which training switches from low to high concurrency when the clients' remaining resource budget drops below a predefined threshold. This method requires prior knowledge of client resources, which might be unknown or difficult to retrieve. In contrast, FedAGF incrementally adjusts concurrency based on the global model's training progress, increasing participation whenever stagnation is detected. By reacting to model evolution rather than following fixed schedules, FedAGF offers a more flexible and efficient solution.

III. METHODOLOGY

A. Preliminaries: Budget-Constrained FL

We consider a centralized, synchronous FL system composed of a single server and energy-constrained clients. We use FedAvg [9] as the backbone of our strategy. Algorithm 1 presents the FedAvg pseudo-code under energy constraints, while Table I summarizes the notations used, where bold denotes vectors, italics denote scalars, and calligraphic letters denote sets.

A central server coordinates a set of devices $\mathcal{S}$, with each device holding a local training dataset $\mathcal{D}_s$. The server initializes the global model $\mathbf{w}_1$ with random weights (line 1). Then, the learning process iterates until no device has sufficient energy to contribute further (lines 3–12). In each round t, the server randomly selects a subset of devices $\mathcal{S}_t \in \mathcal{S}$ to participate in training (line 4). The number of selected devices, i.e., the concurrency $C = |\mathcal{S}_t|$, is a predefined hyperparameter typically fixed over all rounds. Selected devices download the current global model parameters $\mathbf{w}_t$ (line 6), run K iterations of local training on the local dataset $\mathcal{D}_s$ (line 7), and compute the local updates $\mathbf{\Delta}_{s,t}$ (line 8) that are sent to the server (line 9). The server aggregates the collected updates into the "pseudo-gradient" $\mathbf{\Delta}_t$ (line 10), which is used to obtain a new global model version $\mathbf{w}_{t+1}$ (line 11).

TABLE I
FEDAVG NOTATIONS.

	Description
$\mathcal{S}$	Set of devices
$\mathcal{D}_s$	Local dataset of device s
t	Current synchronization round
$\mathcal{S}_t$	Subset of devices participating to round t, $S_t \subset \mathcal{S}$
C	Concurrency level
$\mathbf{w}_t$	Global model weights at round t
K	Number of local training iterations
$\mathbf{w}_{s,t}$	Local model weights of device s at round t
$\mathbf{\Delta}_{s,t}$	Local update of device s at round t
$\mathbf{\Delta}_t$	Global model pseudo-gradient at round t

Under such an energy-constrained formulation, the number of synchronization rounds is limited by the energy budget of clients and is directly affected by the concurrency level C. A higher C results in more participants per round, which increases each client's participation frequency and accelerates budget consumption, thereby reducing the number of possible rounds. A lower C reduces participation frequency, enabling more rounds, but at the cost of slower convergence. With a constant C, FedAvg is constrained by a predefined trade-off between the number of rounds and label coverage, potentially compromising model quality, either because too few updates contribute to the global model or because skew-induced bias results in insufficient class diversity.

To overcome this limitation, we propose to deploy an adaptive strategy able to dynamically regulate concurrency in response to training progress. The resulting FedAGF protocol is detailed in the following sections.

B. Energy Model & Optimization Goal

Following the model introduced in [43], the energy consumption E_s of a client s during a round is given by:

$$E_s = E_s^{\text{train}} + E_s^{\text{sync}}\,, \tag{1}$$

where E_s^{train} represents the energy consumed for the local training (lines 7–8 in Algorithm 1) and E_s^{sync} is the energy spent for the communication with the server (lines 6 and 9 in Algorithm 1). E_s^{train} depends on factors such as device hardware (e.g., CPU model, clock frequency), model complexity, and the number of local training iterations, while E_s^{sync} depends on the communication technology (e.g., WiFi, 5G) and the size of the transmitted payloads.

Over all the rounds, the total energy consumption of a client s is as follows:

$$E_s^{\text{tot}} = \sum_t \mathbb{1}_{s \in \mathcal{S}_t} \cdot E_s\,, \tag{2}$$

with $\mathbb{1}_{s \in \mathcal{S}_t} = 1$ if client s participates in round t, 0 otherwise.

In a realistic cross-device FL deployment, each device is subject to a maximum energy budget $E_s^{\max}$, often defined by restrictions on the battery capacity or user-defined usage limits, which sets the following constraint to be satisfied:

$$E_s^{\text{tot}} = \sum_t \mathbb{1}_{s\in\mathcal{S}_t} \cdot E_s \leq E_s^{\max}, \quad \forall s \in \mathcal{S}\,. \tag{3}$$

Since a precise estimation of E_s for each client is rather complex due to the variety of hardware architectures deployed and the unpredictable operating conditions, including the network settings and load, our formulation relies on a system-independent abstraction where energy is defined through the *participation budget* P ($= E_s^{\max}/E_s$). Since a device consumes energy from the available budget when it participates in a round, the total number of rounds a client may join is bounded by P, which translates the energy constraint into a simpler, hardware-agnostic formulation as follows:

$$\sum_t \mathbb{1}_{s\in\mathcal{S}_t} \cdot E_s \leq E_s^{\max} \Rightarrow \sum_t \mathbb{1}_{s\in\mathcal{S}_t} \leq P\,. \tag{4}$$

We assume uniform budgets across devices, that is, all clients share the same bound P, even if, in reality, clients may have different absolute energy budgets $E_s^{\max}$. This generalization yields a system-independent framework for analyzing budget-constrained FL methods.

According to this abstraction and modeling, the objective of the optimization problem is to maximize the final model accuracy subject to the participation budget:

$$\max \text{ Accuracy} \quad \text{s.t.} \quad \sum_t \mathbb{1}_{s\in\mathcal{S}_t} \leq P,\ \forall s \in \mathcal{S}\,. \tag{5}$$

FedAGF addresses this optimization problem by promoting C as a dynamic control variable. It operates a gradient-aware protocol that monitors the pseudo-gradient $\mathbf{\Delta}_t$ and derives an aggregated proxy signal used as feedback to adjust C according to the evolving training dynamics. This mechanism enables a flexible tradeoff between training effort and label coverage, ensuring that concurrency increases only when additional participation is needed to drive further model improvement. In the next subsections, we describe how FedAGF implements gradient monitoring and concurrency control to realize this adaptive protocol.

C. FedAGF Gradient Monitoring

Training dynamics in FL can be interpreted through the evolution of the global pseudo-gradient, which reflects both the direction and magnitude of model updates across rounds. In an ideal setting, pseudo-gradients are initially large and consistently aligned, driving the model rapidly toward the optimum, and then progressively diminish, eventually oscillating near zero as convergence is reached. In cross-device FL, however, non-idealities such as extreme label skew and partial participation break this pattern. These factors may cause the model to drift and stall in sub-optimal regions, where pseudo-gradients remain far from zero and fail to bring the model closer to the optimum. To detect such stagnant phases, FedAGF introduces a novel gradient-feedback mechanism. Specifically, we derive a scalar proxy signal that digests the progress state of the global pseudo-gradient over rounds into a scalar. This signal suggests whether the training dynamics follow a meaningful descent or have plateaued due to skew-induced inconsistency. To the best of our knowledge, no prior work has proposed such a scalar feedback descriptor for monitoring the evolution of the global pseudo-gradient. FedAGF uses this signal as the trigger for adaptive concurrency control.

Algorithm 2: FedAGF Gradient Monitoring.

Procedure GRADIENTFEEDBACK($\mathbf{\Delta}_t, \tilde{\mathbf{n}}_{t-1}, \tilde{\mathbf{d}}_{t-1}$):
- $\beta = 1 - \frac{2}{\tau+1}$
- $\tilde{\mathbf{n}}_t = (1-\beta)\,\mathbf{\Delta}_t + \beta\,\tilde{\mathbf{n}}_{t-1}$
- $\tilde{\mathbf{d}}_t = (1-\beta)\,|\mathbf{\Delta}_t|_{\text{abs}} + \beta\,\tilde{\mathbf{d}}_{t-1}$
- $\mathbf{n}_t = \tilde{\mathbf{n}}_t / (1-\beta^t)$
- $\mathbf{d}_t = \tilde{\mathbf{d}}_t / (1-\beta^t)$
- $\mathbf{g}_t = |\mathbf{n}_t|_{\text{abs}} / \mathbf{d}_t$
- $g_t^{\text{avg}} \leftarrow$ Element-wise average of $\mathbf{g}_t$
- **return** g_t^{avg}

To operate this mechanism, FedAGF introduces the gradient feedback vector $\mathbf{g}_t$ defined over an observation window of τ rounds as follows:

$$\mathbf{g}_t = \frac{|\sum_{i=t-\tau+1}^{t} \mathbf{\Delta}_i|_{\text{abs}}}{\sum_{i=t-\tau+1}^{t} |\mathbf{\Delta}_i|_{\text{abs}}}\,, \tag{6}$$

where $\mathbf{\Delta}_i$ is the pseudo-gradient at round i, $|\cdot|_{\text{abs}}$ denotes the element-wise absolute value, and the division operator indicates the element-wise division between the numerator and denominator. Each element of $\mathbf{g}_t$ captures the degree of alignment of the corresponding model parameter's pseudo-gradient across the observation window, providing a parameter-wise indicator of learning progress. To obtain an aggregated measure of the overall pseudo-gradient behavior, we compute the average of all elements in $\mathbf{g}_t$, which yields the scalar gradient feedback signal g_t^{avg}.

Intuitively, $g_t^{\text{avg}} \in [0, 1]$ delivers a global indicator of the training status. Values close to one indicate that pseudo-gradients remain consistently aligned across rounds, a behavior typical of early learning phases when the model is making substantial progress. Values close to zero correspond to oscillations around zero, as expected near convergence when updates cancel each other out. Intermediate values suggest unstable dynamics and potential stagnation, often resulting from biased updates. FedAGF leverages this information to regulate the concurrency level, as detailed in Section III-D.

Direct computation of Eq. (6) requires storing τ past pseudo-gradients, which is a memory intensive task. FedAGF avoids this by adopting an exponentially weighted moving average (EMA) approximation, as detailed in Algorithm 2. The procedure maintains two recursive vectors $\tilde{\mathbf{n}}_t$ and $\tilde{\mathbf{d}}_t$ that approximate the numerator and denominator of Eq. (6), enabling efficient online computation of g_t^{avg} with negligible storage overhead.

A well-known issue with EMA estimators is that early values are systematically biased toward the chosen initialization. When the observed pseudo-gradients differ significantly from the initial values, the EMA may require several iterations to reach the correct scale, causing $\tilde{\mathbf{n}}_t$ and $\tilde{\mathbf{d}}_t$ to misestimate the true moving averages. This would distort the gradient-feedback signal and could trigger concurrency adjustments either too early or too late.

To prevent this issue, FedAGF initializes both EMA accumulators to zero ($\tilde{\mathbf{n}}_0 = \mathbf{0}$ and $\tilde{\mathbf{d}}_0 = \mathbf{0}$) and applies standard bias-correction to their uncorrected estimates (lines 5–6 of Algorithm 2), with β^t referring to β raised to the power t. The factor $(1-\beta^t)$ thus evolves as follows: in the early stage $1-\beta^t$ is small, allowing the EMA to track the observed pseudo-gradients; as t increases, $\beta^t \to 0$, and $1-\beta^t \to 1$, making the correction negligible and allowing the EMA to operate as a standard moving average. This ensures that the gradient-feedback signal echoes the true dynamics of the pseudo-gradients throughout the entire training process.

Computing g_t^{avg} requires only element-wise additions, multiplications, and absolute values, which have linear complexity in the number of model parameters. Since these operations are performed at the server side and are easily parallelizable on modern CPUs, the overhead is negligible.

D. FedAGF Concurrency Control Policy

Unlike FedAvg, where the concurrency level C remains fixed throughout training, FedAGF operates C as a dynamic variable C_t that changes over rounds. The control relies on the gradient feedback signal, which enables the system to adapt C_t in response to learning stagnation.

Specifically, FedAGF implements a thresholding policy, described in Algorithm 3. The procedure, hosted by the server and run at the end of each round, takes as input the current concurrency level C_t, the feedback signal g_t^{avg}, and two auxiliary variables: g_{min}, the lowest feedback signal observed so far, and r, a counter of consecutive rounds without significant reduction in g_t^{avg}. The two variables g_{min} and r are initialized to 1 and 0, respectively (omitted in the pseudo-code). The final output is the concurrency level C_{t+1} for the next round.

The adaptive policy increments concurrency if the gradient signal feedback g_t^{avg} remains stable for at least τ consecutive rounds. At each round, if g_t^{avg} decreases by more than a predefined threshold ε compared to g_{min}, g_{min} is updated and the counter r is reset (lines 2–4), meaning training is still making progress and concurrency remains unchanged. Otherwise, r is incremented (line 6). When r exceeds the observation window τ, that is an alert of learning stagnation, concurrency is increased by one unit (lines 7–8). After each increment, g_{min} and r are reset to their initial values (lines 9–10) to ensure a stabilization period of at least τ rounds before further adjustments.

The hyperparameters ε and τ regulate the sensitivity of the control mechanism: ε defines what qualifies as a meaningful decrease in g_t, while τ sets the number of rounds to be tolerated before triggering concurrency increment. Their impact on performance is evaluated in Section V-B1.

Algorithm 3: FedAGF Concurrency Control Policy.

1 **Procedure** UPDATECONCURRENCY(C_t, g_t^{avg}, g_{min}, r)**:**
2 **if** $g_t^{avg} < g_{min} - \varepsilon$ **then**
3 $g_{\text{min}} \leftarrow g_t^{\text{avg}}$;
4 $r \leftarrow 0$;
5 **else**
6 $r \leftarrow r + 1$;
7 **if** $r > \tau$ **then**
8 $C_{t+1} \leftarrow C_t + 1$;
9 $g_{\text{min}} \leftarrow 1$;
10 $r \leftarrow 0$;
11 **return** C_{t+1};

Overall, the FedAGF control logic incurs negligible overhead, as it consists of simple scalar comparisons and state-variable updates.

E. FedAGF Workflow

The pseudo-code in Algorithm 4 presents the workflow with FedAGF, highlighting in gray the main changes applied to the standard FedAvg (Algorithm 1). All changes introduced by FedAGF involve lightweight server-side operations and do not affect client computation or communication.

After the initialization of the global model (line 1), the server sets up the variables required for operating FedAGF: the initial concurrency is set to C_{min} (line 3); $\tilde{\mathbf{n}}_t$ and $\tilde{\mathbf{d}}_t$ store the exponential moving average states for the gradient monitoring mechanism (lines 4–5); and g_{min} and r are the auxiliary variables supporting the concurrency control policy (lines 6–7). Table II summarizes FedAGF variables.

Client selection, local training, and aggregation (lines 8–16) follow standard FedAvg, with one key difference: the number of participating clients C_t is dynamically adjusted in each round according to the observed training dynamics.

After model aggregation, the server computes the gradient feedback signal g_t^{avg} (line 18) using the GRADIENTFEEDBACK routine from Algorithm 2, and updates the concurrency level (line 20) via the UPDATECONCURRENCY procedure defined in Algorithm 3. To control the network usage, concurrency is bounded by a maximum value C_{max} (line 21). The effect of C_{min} and C_{max} on performance is evaluated in Section V-B2

We provide a first empirical demonstration that the gradient feedback signal reliably reflects training progress and can be leveraged as a proxy for controlling concurrency scaling. Figure 2 illustrates the values of g_t^{avg} recorded during the training of a five-layer neural network on the CIFAR-100 dataset, under extreme label skew with $L \leq 5$ (further details in Section IV). The orange curve corresponds to FedAvg with a fixed concurrency of $C = 5$. After an initial rapid drop, g_t^{avg} remains almost constant for the remainder of the training, indicating potential learning stagnation. This is precisely where our method proves its effectiveness. The green curve displays g_t^{avg} obtained using FedAGF, starting with the same initial concurrency of 5. Both curves initially overlap, since both methods operate with identical concurrency settings. However, after approximately 1300 rounds, g_t^{avg} gets stable,

Algorithm 4: FedAGF pseudo-code.

$\mathbf{w}_1 \leftarrow$ Random values
$t \leftarrow 1$
$C_t \leftarrow C_{\text{min}}$
$\tilde{\mathbf{n}}_0 \leftarrow \mathbf{0}$
$\tilde{\mathbf{d}}_0 \leftarrow \mathbf{0}$
$g_{\text{min}} \leftarrow 1$
$r \leftarrow 0$
while *Devices have remaining energy* **do**
  Randomly select a subset $\mathcal{S}_t \subset \mathcal{S}$ of C_t clients from devices with available energy
  /* Training & Synchronization */
  for $s \in \mathcal{S}_t$ **do in parallel**
    DOWNLOAD($\mathbf{w}_t$)
    $\mathbf{w}_{s,t} \leftarrow$ TRAIN($\mathbf{w}_t$, $\mathcal{D}_s$, K)
    $\boldsymbol{\Delta}_{s,t} \leftarrow \mathbf{w}_{s,t} - \mathbf{w}_t$
    UPLOAD($\boldsymbol{\Delta}_{s,t}$)
  /* Global Aggregation */
  $\boldsymbol{\Delta}_t \leftarrow \frac{1}{|\mathcal{S}_t|} \sum_{s \in S_t} \boldsymbol{\Delta}_{s,t}$
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \boldsymbol{\Delta}_t$
  /* FedAGF Gradient Monitoring */
  $g_t^{\text{avg}} \leftarrow$ GRADIENTFEEDBACK($\boldsymbol{\Delta}_t, \tilde{\mathbf{n}}_{t-1}, \tilde{\mathbf{d}}_{t-1}$)
  /* FedAGF Concurrency Control */
  $C_{t+1} \leftarrow$ UPDATECONCURRENCY(C_t, $g_t^{\text{avg}}, g_{\text{min}}, r$)
  $C_{t+1} \leftarrow \min(C_{t+1}, C_{\text{max}})$
  $t \leftarrow t + 1$

triggering the first concurrency increment. Further increments follow in later rounds, and, although they accelerate energy consumption, thereby reducing training duration, they allow the model to escape stagnation by pushing g_t^{avg} toward lower values. In this way, FedAGF makes a more effective use of the available budget, leading to a better tradeoff between energy efficiency and model performance, as demonstrated by the experimental results in Section V.

F. Convergence Analysis

We sketch a convergence proof to demonstrate that FedAGF preserves the convergence properties of FedAvg. The analysis follows the proof of [44], adapted to the case of time-varying (and non-decreasing) concurrency.

Notation: Let N be the total number of clients and T the total number of rounds. The global FL objective is given by:

$$\min_{\mathbf{w} \in \mathbb{R}^d} f(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} f_s(\mathbf{w}), \tag{7}$$

with f_s as the local loss of client s.

At each round t, the server samples (without replacement) a subset $\mathcal{S}_t$ of size C_t. Each selected client downloads the current global model $\mathbf{w}_t$, completes K local SGD steps with learning rate η, and then returns its update to the server. The operator $\|\cdot\|$ denotes the ℓ^2-norm, $\mathbb{E}[\cdot]$ the expected value.

TABLE II
FEDAGF VARIABLES.

	Description
C_t	Concurrency level at round t
$\tilde{\mathbf{n}}_t$	EMA of the pseudo-gradient at round t
$\tilde{\mathbf{d}}_t$	EMA of the pseudo-gradient magnitude at round t
$\mathbf{n}_t$	Bias-corrected version of $\tilde{\mathbf{n}}_t$
$\mathbf{d}_t$	Bias-corrected version of $\tilde{\mathbf{d}}_t$
g_t^{avg}	Gradient feedback signal at round t
g_{min}	FedAGF variable storing the minimum value of g_t
r	Number of consecutive rounds with stable g_{min}
C_{min}	Initial concurrency level
C_{max}	Maximum concurrency level
τ	FedAGF monitoring window
ε	FedAGF threshold

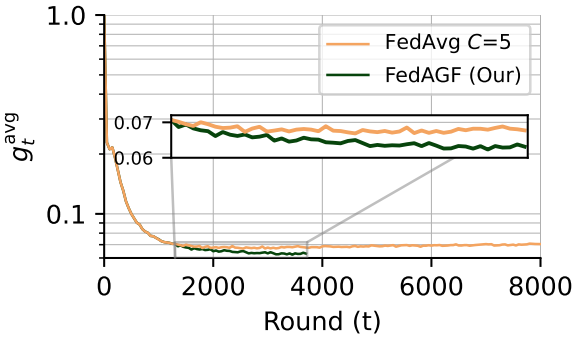


Fig. 2. Gradient feedback signal (g_t^{avg}) during training on CIFAR-100 with $L \leq 5$ [12].

Our analysis is based on the standard assumptions for FL under partial participation and data heterogeneity [44] (Assumptions 1–3), and introduces an additional assumption to formally account for adaptive participation (Assumption 4):

Assumption 1: (L-Smoothness). Each f_s is L-smooth: $\|\nabla f_s(\mathbf{w}) - \nabla f_s(\mathbf{w}')\| \leq L\|\mathbf{w} - \mathbf{w}'\|$, for all $\mathbf{w}, \mathbf{w}'$.

Assumption 2: (Unbiased stochastic gradients with bounded variance). Each local stochastic gradient $\tilde{\nabla} f_s(\mathbf{w})$ satisfies $\mathbb{E}[\tilde{\nabla} f_s(\mathbf{w})] = \nabla f_s(\mathbf{w})$ and $\mathbb{E}\|\tilde{\nabla} f_s(\mathbf{w}) - \nabla f_s(\mathbf{w})\|^2 \leq \sigma^2$.

Assumption 3: (Bounded client dissimilarity). There exists $\zeta \geq 0$ such that $\frac{1}{N} \sum_{s=1}^{N} \|\nabla f_s(\mathbf{w}) - \nabla f(\mathbf{w})\|^2 \leq \zeta^2$ for all $\mathbf{w}$.

Assumption 4: (Adaptive partial participation). At round t, the number of randomly selected clients C_t, with $0 < C_t < N$, never decreases: $C_t \leq C_{t+1}$.

Theorem 1: (Convergence with Adaptive Concurrency). Under assumptions 1–4 and for a suitable client learning rate η, the iterations produced by the training process satisfy the following condition

$$\begin{aligned}\min_{0\le t<T}\mathbb{E}\,\|\nabla f(\mathbf{w}_t)\|^2 \le\ & \underbrace{\mathcal{O}\left(\frac{f(\mathbf{w}_0)-f^\star}{\eta\,K\,T}\right)}_{\text{Initialization error}} \\ &+ \underbrace{\mathcal{O}\left(\eta^2L^2(K-1)\,\sigma^2+\ \eta\,L\,\sigma^2\frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{C_t}\right)}_{\text{Stochastic gradient error}} \\ &+ \underbrace{\mathcal{O}\left(\eta\,K\,L\,\frac{N-\frac{1}{T}\sum_{t=0}^{T-1}C_t}{(N-1)}\,\frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{C_t}\,\zeta^2\right)}_{\text{Partial participation error}} \\ &+ \underbrace{\mathcal{O}\left(\eta^2\,L^2\,K\,(K-1)\,\zeta^2\right)}_{\text{Data heterogeneity error}}.\end{aligned} \tag{8}$$

Proof Sketch: The proof is the same as in [44], where the constant concurrency C is replaced by the time-varying concurrency C_t proper of FedAGF. The one-step expected descent satisfies

$$\begin{aligned}\mathbb{E}[f(\mathbf{w}_{t+1})-f(\mathbf{w}_t)] \le\ & -\frac{\eta\,K}{4}\,\|\nabla f(\mathbf{w}_t)\|^2 \\ &+\ 2\,\eta^3\,K\,L^2(K-1)\big[\sigma^2+4\,K\,\zeta^2\big] \\ &+\ \frac{\eta^2\,K\,L\,\sigma^2}{C_t}\ +\ 3\,\eta^2\,K^2L\,\frac{N-C_t}{(N-1)\,C_t}\,\zeta^2.\end{aligned} \tag{9}$$

Summing over $t=0,...,T-1$ and choosing a sufficiently small η yields

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\,\|\nabla f(\mathbf{w}_t)\|^2 \le\ & \frac{4\big(f(\mathbf{w}_0)-f^\star\big)}{\eta\,K\,T} \\ &+\ 8\,\eta^2L^2(K-1)\big[\sigma^2+4K\zeta^2\big]\ +\ 4\,\eta L\,\sigma^2\,\frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{C_t} \\ &+\ 12\,\eta\,K\,L\,\frac{N-\frac{1}{T}\sum_{t=0}^{T-1}C_t}{N-1}\,\frac{1}{T}\sum_{t=0}^{T-1}\frac{1}{C_t}\,\zeta^2,\end{aligned} \tag{10}$$

which matches the claimed bound after simplifying constants and rearranging the terms.

Discussion: The derived bound shows that adaptive concurrency preserves the convergence guarantees of FedAvg, providing a theoretical foundation to FedAGF.

The error breakdown also offers insight into the training dynamics induced by increasing concurrency, and motivates the working principle of the FedAGF control mechanism. As the term $\frac{1}{T}\sum_{t=0}^{T-1}1/C_t$ decreases with larger C_t, the stochastic-gradient error and the partial-participation error reduce accordingly. Involving more clients reduces bias and variance in the aggregated update. On the other hand, a large C_t accelerates the rate at which the available budget is consumed, reducing the total number of rounds T. As the term $\mathcal{O}(\frac{1}{T})$ grows with smaller T, the initialization error gets larger. Intuitively, fewer rounds lead to poor late-stage refinements.

Effective training, therefore, requires a proper balance between these contrasting effects. The plots in Fig. 1 provide empirical evidence of this tradeoff. Increasing concurrency from $C=5$ to $C=10$ improves accuracy as the stochastic-gradient and partial-participation errors decrease; however, using a too large concurrency $C=20$, the initialization error becomes the dominant contribution and the final accuracy is degraded. FedAGF addresses this trade-off by adjusting concurrency based on the gradient-feedback signal. It maintains low concurrency during early training to preserve participation budgets and increases it only when learning stagnation is detected, allocating additional resources when they yield the greatest benefit.

IV. Experimental Setup

A. Benchmarks & Data Distributions

The experimental evaluation is conducted on the CIFAR-10 and CIFAR-100 datasets [45], using the standard training–test splits. To simulate a cross-device FL scenario, the 50,000 training images were partitioned into $|\mathcal{S}|$ non-overlapping subsets, each assigned to one client. Unless otherwise noted, $|\mathcal{S}|=100$. Following the partitioning strategy of [9], each client dataset is restricted to at most L classes. This is achieved by creating $|\mathcal{S}|\times L$ single-label data shards and randomly distributing them across clients, assigning L shards each. The experiments cover different levels of heterogeneity: $L\le 4$ and $L\le 3$ for CIFAR-10, and $L\le 5$ and $L\le 3$ for CIFAR-100.

B. Model Architecture & Training Hyperparameters

The model architecture is the five-layer convolutional neural network introduced in [9]. In each round, selected clients perform local training for $K=20$ iterations (unless stated otherwise) using Stochastic Gradient Descent (SGD) with learning rate $1\mathrm{e}{-2}$, momentum 0.9, weight decay $1\mathrm{e}{-3}$, and batch size 50.

C. Baselines & Experimental Details

FedAGF is benchmarked against four representative strategies: FedAvg [9], FedRS [13], FedLC [14], and AdaFL [16].

- **FedAvg** is the standard FL algorithm with fixed concurrency C and cross-entropy loss for local training.
- **FedRS** applies a modified softmax operator for local training to reduce imbalance due to missing labels. We grid-searched its hyperparameter in $\{0.5, 0.9\}$.
- **FedLC** proposes to rescale predictions based on local distributions during local training. We grid-searched its hyperparameter in $\{0.1, 0.5\}$.
- **AdaFL** follows a dynamic participation strategy, incrementing concurrency by one every ΔT rounds until a maximum value is reached. We used 5 initial participants, incremented up to 30, and evaluated $\Delta T\in\{100, 200\}$.

For FedAvg, FedRS, and FedLC, C is constant, and the values chosen to explore the tradeoff between participation and synchronization rounds are $\{5, 10, 20\}$. For our proposed FedAGF, the hyperparameters are as follows: $\tau=100$, $\varepsilon=1\mathrm{e}{-4}$, $C_{\min}=5$, $C_{\max}=30$.

These baselines allow us to analyze the effects of different approaches: FedRS and FedLC implement algorithmic adjustments at the local training level, while AdaFL is a dynamic

TABLE III
PERFORMANCE COMPARISONS ON CIFAR-10.

L	Method	$P = 200$	$P = 400$	$P = 600$
$\leq$**4**	**FedAGF** (Our)	**79.29**	**80.51**	**81.22**
	FedAvg C=5	77.82 (-1.48%)	79.53 (-0.99%)	79.57 (-1.65%)
	FedAvg C=10	76.68 (-2.61%)	79.62 (-0.89%)	80.22 (-1.00%)
	FedAvg C=20	76.01 (-3.29%)	78.92 (-1.59%)	79.77 (-1.45%)
	FedRS	78.33 (-0.97%)	78.43 (-2.08%)	78.60 (-2.62%)
	FedLC	78.17 (-1.12%)	78.35 (-2.16%)	78.86 (-2.36%)
	AdaFL	77.83 (-1.46%)	79.61 (-0.90%)	80.29 (-0.93%)
$\leq$**3**	**FedAGF** (Our)	**77.42**	**80.10**	**80.21**
	FedAvg C=5	77.21 (-0.21%)	79.24 (-0.86%)	79.25 (-0.97%)
	FedAvg C=10	75.45 (-1.96%)	77.96 (-2.14%)	79.18 (-1.03%)
	FedAvg C=20	73.45 (-3.97%)	77.15 (-2.95%)	78.88 (-1.33%)
	FedRS	76.75 (-0.66%)	77.15 (-2.95%)	77.87 (-2.34%)
	FedLC	77.09 (-0.33%)	77.33 (-2.77%)	77.47 (-2.74%)
	AdaFL	77.18 (-0.23%)	78.83 (-1.27%)	79.37 (-0.84%)

L: # Labels. C: Concurrency. P: Participation Budget.
Note: The best results are in bold.

TABLE IV
PERFORMANCE COMPARISONS ON CIFAR-100.

L	Method	$P = 200$	$P = 400$	$P = 600$
$\leq$**5**	**FedAGF** (Our)	**43.95**	**46.00**	**46.85**
	FedAvg C=5	38.86 (-5.09%)	41.13 (-4.87%)	41.89 (-4.95%)
	FedAvg C=10	39.03 (-4.92%)	43.18 (-2.82%)	44.52 (-2.33%)
	FedAvg C=20	37.23 (-6.72%)	42.65 (-3.35%)	44.56 (-2.28%)
	FedRS	39.26 (-4.69%)	39.49 (-6.52%)	39.58 (-7.27%)
	FedLC	35.56 (-8.39%)	35.96 (-10.04%)	35.78 (-11.07%)
	AdaFL	39.58 (-4.37%)	44.23 (-1.78%)	44.89 (-1.96%)
$\leq$**3**	**FedAGF** (Our)	**42.26**	**45.32**	**46.36**
	FedAvg C=5	37.41 (-4.84%)	40.37 (-4.95%)	41.41 (-4.96%)
	FedAvg C=10	37.36 (-4.90%)	42.37 (-2.95%)	43.76 (-2.61%)
	FedAvg C=20	33.80 (-8.46%)	41.05 (-4.27%)	43.63 (-2.74%)
	FedRS	35.30 (-6.96%)	35.44 (-9.88%)	36.32 (-10.04%)
	FedLC	32.19 (-10.07%)	31.73 (-13.59%)	32.22 (-14.14%)
	AdaFL	37.96 (-4.30%)	42.67 (-2.65%)	44.14 (-2.22%)

L: # Labels. C: Concurrency. P: Participation Budget.
Note: The best results are in bold.

concurrency strategy alternative to our method. The experimental goal is twofold: (i) determine whether optimization of the local training or of resource management achieves superior performance under skewed data distributions; (ii) assess whether the gradient-aware protocol in FedAGF outperforms periodic concurrency increments as in AdaFL.

D. Budget Constraints & Evaluation Metrics

Different constraints are considered, which cover different FL scenarios with an increasing participation budget $P \in \{200, 400, 600\}$[1]. To assess the global model performance, the averaged test accuracy is measured over the latest 30 rounds before the participation budget is met to filter out inherent oscillations during training [46].

V. RESULTS

A. Performance Comparisons

Tables III and IV report the results of the comparison between FedAGF and the selected baselines for the CIFAR-10 and CIFAR-100 datasets, respectively. For FedRS, FedLC, and AdaFL, we report the best accuracy achieved after hyperparameter and concurrency tuning.

As expected, consistently with prior work [13], [14], accuracy levels on CIFAR-100 are significantly lower than CIFAR-10 due to more classes and fewer samples per class.

FedAGF consistently outperforms all baselines for every benchmark and constraint under analysis. On CIFAR-10, accuracy gains range from +0.21% to +3.97%, while on the more challenging CIFAR-100, improvements are even larger: +2.28% to +8.46% over FedAvg, +4.69% to +10.04% over FedRS, +8.39% to +14.14% over FedLC, and +1.78% to +4.37% over AdaFL. These results demonstrate the advantage of our gradient-aware concurrency policy and its ability to adapt to datasets with varying complexities, optimizing the resource usage accordingly.

[1]With a fixed concurrency $C = 5$, these budgets correspond to 4,000, 8,000, and 12,000 synchronization rounds, respectively.

The results on FedAvg highlight the limits of static concurrency. The concurrency level that maximizes accuracy depends on both the label skew L and the participation budget P, with no universally optimal choice. For instance, on CIFAR-10 with $L \leq 4$, the best accuracy is obtained with $C = 5$ (77.82%) for $P = 200$, but with $C = 10$ (80.22%) for $P = 600$. The lack of adaptability makes FedAvg difficult to tune in practice, while FedAGF automatically adjusts concurrency based on the training progress, providing self-adaptation to different settings and constraints.

The modified local training procedures of FedRS and FedLC show more robust learning under extreme label skew, but often reduce long-term plasticity in acquiring new knowledge. On CIFAR-100 with $L \leq 3$, FedLC accuracy improves by only +0.03% (from 32.19% to 32.22%) when the budget triples ($P = 200$ to $P = 600$), while FedAGF achieves a +4.10% increase (from 42.26% to 46.36%) with the same setting. This suggests that algorithmic changes to local training may help initial convergence but fail to sustain improvements when more budget is available.

AdaFL introduces dynamic concurrency but relies on fixed-interval increments, which ignore the actual training dynamics and can therefore lead to suboptimal resource usage. For instance, on CIFAR-10 with $L \leq 4$ and $P = 200$, AdaFL (77.83%) underperforms both FedRS (78.33%) and FedLC (78.17%). A deeper analysis of the training logs reveals that AdaFL triggers premature concurrency increments, consuming the participation budget before reaching peak accuracy. FedAGF, instead, increases concurrency only when progress stalls, ensuring a more effective allocation of resources. Figure 3 illustrates this difference by comparing how concurrency varies over rounds during the training on CIFAR-100 with $L \leq 5$ and $L \leq 3$. In AdaFL, concurrency follows a predefined schedule dictated entirely by its hyperparameter ΔT, and is therefore identical across both experiments, regardless of the degree of label skew. This rigid scheme highlights two drawbacks: concurrency may increase too early, when not actually needed, and it cannot adapt to different data distributions. By contrast, FedAGF shows concurrency curves that vary with L,

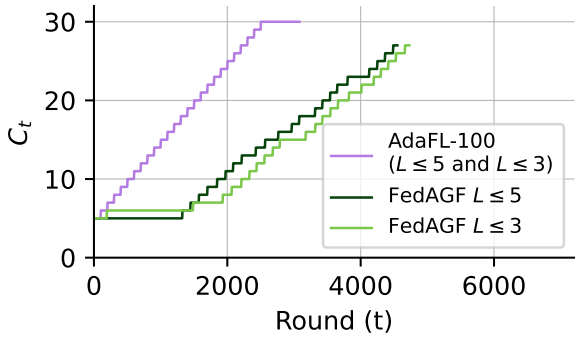


Fig. 3. Concurrency evolution (C_t) during training on CIFAR-100 with AdaFL and FedAGF.

adapting to heterogeneous conditions thanks to increments that can happen at irregular intervals, based on the actual training dynamics. Such adaptability is key to achieving consistent performance improvements across different settings.

B. Sensitivity Analysis to FedAGF Hyperparameters

1) Sensitivity to τ and ε: In the previous comparisons, we evaluated FedAGF with a fixed hyperparameter setting: $\tau = 100$ and $\varepsilon = 1\text{e}{-4}$. We now investigate the sensitivity to these parameters. Specifically, we explore different configurations with $\tau \in \{100, 150, 200\}$ and $\varepsilon \in \{1\text{e}-3, 1\text{e}{-4}, 5\text{e}{-4}, 1\text{e}-5\}$. Tables V and VI collect a subset of experiments on CIFAR-100, considering two degrees of label skew: $L \leq 5$ and $L \leq 3$. The results show that FedAGF is highly robust to variations in the hyperparameters, ensuring consistent performance across all settings. The largest observed differences are 2.19% for $L \leq 5$ (41.88 vs. 44.07 at $P = 200$) and 2.05% for $L \leq 3$ (45.61 vs. 47.66 at $P = 600$). Notably, every configuration still surpasses the best results achieved by the baselines in Section V-A, confirming that FedAGF delivers reliable improvements even when τ and ε are not perfectly tuned.

Between the two hyperparameters, ε has a more marginal effect on performance, while a proper finetuning of τ is desirable to reach larger improvements. For example, on CIFAR-100 with $L \leq 5$ and $P = 200$ (Table V), setting $\tau = 150$ improves accuracy by +2.29% compared to $\tau = 200$ (44.07 vs. 41.78) for the same value of $\varepsilon = 1\text{e}{-3}$. Interestingly, the results suggest that the optimal choice of τ mainly depends on the available resources. With lower budgets ($P = 200$), smaller values of τ are more effective, as concurrency increments are triggered earlier, allowing higher participation before the full budget is depleted. Conversely, with larger budgets ($P = 400$ and $P = 600$), higher values of τ are preferable, as delayed concurrency scaling allows more rounds at lower energy cost each, namely, a longer training.

2) Sensitivity to C_{min} and C_{max}: This subsection investigates the impact on performance of the initial concurrency level C_{min} and the maximum concurrency level C_{max}, which are key parameters to consider. A lower C_{min} reduces label coverage per round but allows more conservative resource usage in the early training phases, whereas a higher C_{min} increases coverage at the cost of faster budget consumption.

TABLE V
FEDAGF PERFORMANCE UNDER DIFFERENT VALUES OF MONITORING WINDOW (τ) AND THRESHOLD (ε) ON CIFAR-100 WITH $L \leq 5$.

τ	ε	$P = 200$	$P = 400$	$P = 600$
100	**1e−3**	43.71	46.39	46.93
	5e−4	43.35	46.04	46.96
	1e−4	43.95	46.00	46.85
	1e−5	43.85	45.80	47.13
150	**1e−3**	44.07	46.28	47.36
	5e−4	43.95	46.40	47.33
	1e−4	43.84	45.66	47.31
	1e−5	43.50	46.57	47.21
200	**1e−3**	41.88	46.21	46.89
	5e−4	43.02	46.57	47.11
	1e−4	42.55	46.64	47.69
	1e−5	42.55	46.29	47.47

TABLE VI
FEDAGF PERFORMANCE UNDER DIFFERENT VALUES OF MONITORING WINDOW (τ) AND THRESHOLD (ε) ON CIFAR-100 WITH $L \leq 3$.

τ	ε	$P = 200$	$P = 400$	$P = 600$
100	**1e−3**	40.95	44.48	45.61
	5e−4	41.45	45.12	46.11
	1e−4	42.26	45.32	46.36
	1e−5	41.72	45.26	46.51
150	**1e−3**	42.02	46.11	46.85
	5e−4	41.54	45.59	47.10
	1e−4	41.18	45.41	47.34
	1e−5	41.77	46.20	47.59
200	**1e−3**	41.31	45.97	47.66
	5e−4	41.84	46.23	47.39
	1e−4	41.45	46.45	47.45
	1e−5	40.66	45.97	47.09

Since the maximum concurrency C_{max} is constrained by system factors such as network capacity or client availability, a too large value may be impractical in real deployments.

Tables VII and VIII report the results for different configurations with $C_{\text{min}} \in \{3, 5\}$ and $C_{\text{max}} \in \{20, 30\}$, evaluated on CIFAR-100 with $L \leq 5$ and $L \leq 3$, respectively. FedAGF still proves its robustness across different settings, with just minor variations ranging from 0.21% (45.11 vs. 45.32 at $P = 400$ for $L \leq 3$) to 1.41% (46.02 vs. 47.43 at $P = 600$ for $L \leq 5$). More importantly, each configuration achieves higher accuracy than the baselines from Section V-A. For instance, on CIFAR-100 with $L \leq 5$ and $P = 600$, the worst-performing configuration of FedAGF (46.02%) still outperforms FedAvg by +1.46% (44.56%), FedRS by +6.44% (39.58%), FedLC by +10.23% (35.79%), and AdaFL by +1.13% (44.89%). These findings reveal that FedAGF maintains its advantage even under varying concurrency bounds, demonstrating its practical applicability across diverse deployment scenarios.

C. Scalability Analysis

This section presents additional insights that further prove the robustness and scalability of FedAGF under different system configurations. Specifically, it first analyzes the behavior of FedAGF against the number of local training iterations K, then it assesses the effects of different system sizes, that

TABLE VII
FEDAGF PERFORMANCE UNDER DIFFERENT VALUES OF INITIAL CONCURRENCY (C_{MIN}) AND MAXIMUM CONCURRENCY (C_{MAX}) ON CIFAR-100 WITH $L \leq 5$.

$\mathbf{C_{min}}$	$\mathbf{C_{max}}$	$P = 200$	$P = 400$	$P = 600$
3	**20**	43.42	46.40	46.51
	30	43.42	46.58	47.43
5	**20**	43.95	45.25	46.02
	30	43.95	46.00	46.85

TABLE VIII
FEDAGF PERFORMANCE UNDER DIFFERENT VALUES OF INITIAL CONCURRENCY (C_{MIN}) AND MAXIMUM CONCURRENCY (C_{MAX}) ON CIFAR-100 WITH $L \leq 3$.

$\mathbf{C_{min}}$	$\mathbf{C_{max}}$	$P = 200$	$P = 400$	$P = 600$
3	**20**	42.01	45.11	45.63
	30	42.01	45.11	46.73
5	**20**	42.26	45.15	45.36
	30	42.26	45.32	46.36

is, the number of federated clients. The analysis is done by means of a comparison between FedAvg with $C = 10$, which is the best-performing configuration observed, and FedAGF with $\tau = 100$, $\varepsilon = 1\text{e}{-4}$, $C_{\text{min}} = 5$, and $C_{\text{max}} = 30$; all results are for CIFAR-100 with $L \leq 5$ and $L \leq 3$.

1) Impact of Local Training Iterations K: The value of K serves as the main knob controlling the duration of each local training: a larger K increases the amount of computation each client performs locally, extending the round duration; a smaller K shortens the duration of each round but may require more rounds to reach convergence. Since FL servers impose synchronization deadlines to avoid stragglers and ensure overall training efficiency, tuning K according to clients' hardware capabilities enables a balance between per-round computation and global training time. In other words, K is a system-level parameter against which any FL algorithm must guarantee stability. To assess FedAGF's robustness under different round durations, we evaluated $K \in \{20, 30, 40\}$.

Tables IX and X report the test accuracy for each K for $L \leq 5$ and $L \leq 3$, respectively, and considering participation budgets $P \in \{200, 400, 600\}$. FedAGF outperforms FedAvg for all settings and label distributions explored. Gains range from 2.33% (at $P = 600$ with $K = 20$) to 4.92% (at $P = 200$ with $K = 20$) for $L \leq 5$, and from 2.61% (at $P = 600$ with $K = 20$) to 4.90% (at $P = 200$ with $K = 20$) for $L \leq 3$. These results further highlight the flexibility of FedAGF, which ensures comparable improvements across multiple values of local iterations, even under many degrees of label skew. Moreover, FedAGF shows substantially greater invariance with respect to variations in K compared to FedAvg. For example, with $L \leq 5$ (Table IX), FedAvg accuracy fluctuates by up to 1.96% (from 41.80% to 43.76% at $P = 600$), whereas FedAGF varies only by 0.77% in the worst case (from 47.10% to 46.85% at $P = 600$). This reduced sensitivity highlights that FedAGF can effectively adapt to different learning dynamics induced by different K values, ensuring steady model progress regardless of the local training duration.

TABLE IX
FEDAVG VS. FEDAGF PERFORMANCE UNDER DIFFERENT NUMBER OF LOCAL TRAINING ITERATIONS (K) ON CIFAR-100 WITH $L \leq 5$.

K	Method	$P = 200$	$P = 400$	$P = 600$
20	**FedAGF**	43.95 (+4.92%)	46.00 (+2.82%)	46.85 (+2.33%)
	FedAvg	39.03	43.18	44.52
30	**FedAGF**	44.07 (+3.79%)	46.56 (+3.56%)	47.62 (+3.96%)
	FedAvg	40.28	43.00	43.66
40	**FedAGF**	44.51 (+4.27%)	46.53 (+4.03%)	47.10 (+3.37%)
	FedAvg	40.24	42.50	43.73

TABLE X
FEDAVG VS. FEDAGF PERFORMANCE UNDER DIFFERENT NUMBER OF LOCAL TRAINING ITERATIONS (K) ON CIFAR-100 WITH $L \leq 3$.

K	Method	$P = 200$	$P = 400$	$P = 600$
20	**FedAGF**	42.26 (+4.90%)	45.32 (+2.95%)	46.36 (+2.61%)
	FedAvg	37.36	42.37	43.76
30	**FedAGF**	41.99 (+3.25%)	44.88 (+3.41%)	46.28 (+4.15%)
	FedAvg	38.74	41.47	42.13
40	**FedAGF**	42.13 (+3.49%)	44.95 (+3.96%)	46.09 (+4.29%)
	FedAvg	38.64	40.99	41.80

2) Impact of Client Numbers $|\mathcal{S}|$: For this study we consider three increasing system sizes, $|\mathcal{S}| \in \{100, 500, 1000\}$, while keeping all other settings untouched. Tables XI and XII present the resulting test accuracies for $L \leq 5$ and $L \leq 3$, respectively, considering the set of participation budgets $P \in \{200, 400, 600\}$. No matter the system scale, FedAGF always reaches higher accuracy, with gains ranging from +1.63% (at $P = 600$ with $|\mathcal{S}| = 500$) to +4.92% (at $P = 200$ with $|\mathcal{S}| = 100$) for $L \leq 5$, and +1.38% (at $P = 600$ with $|\mathcal{S}| = 1000$) to +4.90% (at $P = 200$ with $|\mathcal{S}| = 100$) for $L \leq 3$. Performance improvements are greatest for $|\mathcal{S}| = 100$, but are still positive and stable as clients grow, demonstrating that FedAGF scales effectively with system size.

D. Impact of Gradient Compression

The continued synchronization between clients and server makes communication a major bottleneck in cross-device FL, especially in IoT scenarios with scarce bandwidth or strict data usage limits. Gradient compression methods are often deployed to reduce the volume of exchanged data without affecting, or with minimal impact on, the accuracy. It is therefore important to evaluate whether the monitoring mechanism and control policy of FedAGF remain effective in this context. We focus on quantization as a representative and relevant compression technique. Specifically, we test FedAGF with the low-precision training scheme introduced in [23], which can achieve substantial savings even under heterogeneous data. We compare standard 32-bit full-precision training with two quantized configurations at 8-bit and 6-bit precision. Tables XIII and XIV collect the results on CIFAR-100 for $L \leq 5$ and $L \leq 3$, with participation budget $P \in \{200, 400, 600\}$.

In line with the results reported in [23], the training at 8 bits achieves an accuracy close to full precision, whereas at 6-bit

TABLE XI
FEDAVG VS. FEDAGF PERFORMANCE UNDER DIFFERENT CLIENT NUMBERS ($|\mathcal{S}|$) ON CIFAR-100 WITH $L \leq 5$.

$\|\mathcal{S}\|$	Method	$P = 200$	$P = 400$	$P = 600$
100	**FedAGF**	43.95 (+4.92%)	46.00 (+2.82%)	46.85 (+2.33%)
	FedAvg	39.03	43.18	44.52
500	**FedAGF**	47.40 (+1.75%)	47.94 (+1.91%)	47.84 (+1.63%)
	FedAvg	45.65	46.03	46.21
1000	**FedAGF**	48.66 (+2.29%)	48.53 (+2.16%)	48.69 (+2.11%)
	FedAvg	46.37	46.37	46.58

TABLE XII
FEDAVG VS. FEDAGF PERFORMANCE UNDER DIFFERENT CLIENT NUMBERS ($|\mathcal{S}|$) ON CIFAR-100 WITH $L \leq 3$.

$\|\mathcal{S}\|$	Method	$P = 200$	$P = 400$	$P = 600$
100	**FedAGF**	42.26 (+4.90%)	45.32 (+2.95%)	46.36 (+2.61%)
	FedAvg	37.36	42.37	43.76
500	**FedAGF**	47.04 (+2.02%)	47.63 (+2.36%)	47.45 (+1.73%)
	FedAvg	45.02	45.27	45.72
1000	**FedAGF**	46.29 (+2.11%)	46.86 (+1.76%)	47.16 (+1.38%)
	FedAvg	44.18	45.10	45.78

TABLE XIII
FEDAVG VS. FEDAGF PERFORMANCE ($C = 10$) WITH LOW-PRECISION TRAINING [23] UNDER DIFFERENT BITWIDTHS (BITS) ON CIFAR-100 WITH $L \leq 5$.

Bits	Method	$P = 200$	$P = 400$	$P = 600$
32	**FedAGF**	43.95 (+4.92%)	46.00 (+2.82%)	46.85 (+2.33%)
	FedAvg	39.03	43.18	44.52
8	**FedAGF**	42.10 (+4.15%)	45.26 (+4.01%)	46.29 (+3.76%)
	FedAvg	37.95	41.52	42.53
6	**FedAGF**	30.54 (+5.13%)	31.42 (+7.50%)	31.37 (+10.42%)
	FedAvg	25.41	23.92	20.95

TABLE XIV
FEDAVG VS. FEDAGF PERFORMANCE ($C = 10$) WITH LOW-PRECISION TRAINING [23] UNDER DIFFERENT BITWIDTHS (BITS) ON CIFAR-100 WITH $L \leq 3$.

Bits	Method	$P = 200$	$P = 400$	$P = 600$
32	**FedAGF**	42.26 (+4.90%)	45.32 (+2.95%)	46.36 (+2.61%)
	FedAvg	37.36	42.37	43.76
8	**FedAGF**	40.05 (+4.31%)	44.37 (+3.37%)	45.62 (+3.42%)
	FedAvg	35.74	41.00	42.20
6	**FedAGF**	32.54 (+5.44%)	32.68 (+5.88%)	32.89 (+7.15%)
	FedAvg	27.10	26.80	25.74

the loss increases, both for FedAvg and FedAGF. However, FedAGF outperforms FedAvg across all settings. With 8 bits the accuracy gains range from 3.76% (at $P = 600$) to 4.15% (at $P = 200$) for $L \leq 5$, and from 3.42% (at $P = 600$) to 4.31% (at $P = 200$) for $L \leq 3$. With 6 bits, the gains increase (from +5.13% to +10.42%), suggesting the adaptive mechanism of FedAGF is even more effective at lower precisions, when updates become noisy. It is worth noting that FedAGF at 8 bits beats FedAvg at full 32-bit precision, demonstrating not just its resilience to quantization errors but also its potential role in mitigating the loss introduced by gradient compression methods. For instance, when $L \leq 5$, FedAGF achieves 40.05% vs. 37.36% (at $P = 200$), 44.37% vs. 42.37% (at $P = 400$), and 45.62% vs. 43.76% (at $P = 600$).

Moreover, FedAvg exhibits convergence instability under aggressive quantization, with accuracy decreasing as the participation budget increases. In Table XIV, FedAvg at 6 bits drops from 27.10% at $P = 200$ to 25.74% at $P = 600$, while FedAGF maintains higher and increasing accuracy between 32.54% and 32.89%, retaining performance as P grows.

E. Impact of Device Heterogeneity

All previous experiments assume a uniform participation budget across devices to enable fair comparisons with existing mitigation techniques for extreme label skew, which are designed and evaluated without considering device heterogeneity. This ensures that any performance variation is attributable to the algorithms' ability to handle skewed data rather than to side effects arising from device diversity.

However, real-world cross-device FL deployments rarely satisfy this assumption. Mobile and IoT devices often differ in computing capability, battery capacity, and availability, resulting in non-uniform participation budgets. To evaluate FedAGF in such environments, we follow the procedure in [47] and group clients into three categories with different participation budgets: weak ($P = 200$), medium ($P = 400$), and strong ($P = 600$). Operating 100 clients, we consider two representative distributions: (i) a balanced split with a ratio of 34:33:33 (34 weak, 33 medium, and 33 strong), and (ii) a highly skewed split dominated by weak devices with a ratio of 80:10:10 (80 weak, 10 medium, and 10 strong).

Table XV reports the test accuracy of FedAGF and FedAvg ($C = 10$) on CIFAR-100 for label-skew levels $L \leq 5$ and $L \leq 3$. Across all settings, FedAGF achieves the highest accuracy with gains up to 4.55%. As the proportion of weak devices increases, both methods experience performance degradation, yet with remarkably different sensitivity. FedAvg shows larger drops indeed; for instance, accuracy reduces by 2.02% (from 43.12% to 41.10%) for $L \leq 5$, and 3.66% (from 42.57% to 38.91%) for $L \leq 3$. FedAGF gets more stable instead, with reductions of just 1.56% (from 46.45% to 44.89%) for $L \leq 5$, and 1.63% (from 45.09% to 43.46%) for $L \leq 3$. When participation is static, device heterogeneity exacerbates the effects of label skew, whereas adaptive participation maintains consistent performance even under highly imbalanced device capabilities. These results show that adaptive participation becomes even more important in real-world FL deployments.

F. Limitations and Future Work

Our experimental evaluation shows two main limitations. First, the experimental campaign conducted on CIFAR-10 and CIFAR-100 might not have fully captured the diversity and scale of real-world IoT data. However, these datasets are a de facto standard in the FL literature. Their adoption enables fair comparisons with prior work and reproducibility for future investigations. They were also sufficiently complex to reproduce the effects of label skew, which was the

TABLE XV
FEDAVG VS. FEDAGF PERFORMANCE ON CIFAR-100 USING DIFFERENT DISTRIBUTIONS OF WEAK ($P = 200$), MEDIUM ($P = 400$), AND STRONG ($P = 600$) DEVICES.

L	Method	Device Proportions 34:33:33	Device Proportions 80:10:10
≤ 5	**FedAGF**	46.45 (+3.33%)	44.89 (+3.79%)
	FedAvg	43.12	41.10
≤ 3	**FedAGF**	45.09 (+2.52%)	43.46 (+4.55%)
	FedAvg	42.57	38.91

focus of our study. Second, our analysis was hosted in a controlled, simulation-based environment rather than on a real IoT network. Although we acknowledge that a deployment on physical IoT devices fed with realistic online data may introduce additional challenges, our choice was intentional: simulations allowed us to systematically isolate the joint effects of label skews, adaptive concurrency, and heterogeneous energy constraints. Running the study on real infrastructure would have made the evaluation highly system-specific and difficult, if not impossible, to control and reproduce due to variability in hardware architectures, network operating conditions, background workloads, and traffic patterns. Similar considerations motivated the adoption of the participation-budget model, which replaced direct energy measurements with a hardware-agnostic proxy.

In light of these considerations, our experimental setup provided a reliable testbed that mirrored essential characteristics of real-world environments characterized by extreme label skew, partial client participation, constrained participation budgets, and heterogeneous device capabilities, while still ensuring reproducibility and alignment with established practices in the literature for fair comparisons.

We plan to extend our evaluation to a real IoT infrastructure. The focus will shift from a pure state-of-the-art comparison to complementing our simulation-driven findings with real-system evidence. We intend to integrate FedAGF into a physical IoT infrastructure and conduct experiments with on-device energy profiling across heterogeneous device fleets.

VI. CONCLUSION

FedAGF is a new gradient-aware protocol that redefines concurrency as a dynamic variable in FL. Unlike standard methods that rely on static participation, FedAGF introduces a monitoring mechanism that tracks a gradient signal to detect stagnation during the global model training and adapts client concurrency accordingly. Such an adaptive policy enables the FL framework to allocate resources more effectively, increasing participation only when it brings tangible gains.

The experimental results demonstrate that FedAGF leads to substantial benefits. Compared to the state-of-the-art, FedAGF achieves consistent accuracy improvement of up to 14.14%, while keeping robustness to hyperparameter variations, local training configurations, and system size.

Overall, our findings highlight that concurrency management is key in enabling efficient and scalable FL, and FedAGF is a practical option to maximize model quality while meeting resource constraints.

REFERENCES

[1] H. Woisetschläger, A. Erben, B. Marino, S. Wang, N. D. Lane, R. Mayer, and H. Jacobsen, "Federated learning priorities under the european union artificial intelligence act," *arXiv:2402.05968*, 2024.

[2] T. H. Hsu, H. Qi, and M. Brown, "Federated visual classification with real-world data distribution," in *Proc. Eur. Conf. Comp. Vis (ECCV)*, Aug. 2020, pp. 76—92.

[3] L. Wulfert, C. Wiede, and A. Grabmaier, "Tinyfl: On-device training, communication and aggregation on a microcontroller for federated learning," in *Proc. 21st IEEE Interregional NEWCAS Conf. (NEWCAS)*, Jun. 2023, pp. 1–5.

[4] D. Kushwaha, M. Kalavadia, V. Hegde, and O. J. Pandey, "Energy-efficient and latency-aware blockchain-enabled federated learning for edge networks," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 71, no. 3, pp. 1126–1130, Mar. 2024.

[5] H. Ding, Y. Yan, Y. Lu, J. Xue, and H. Wang, "Uncertainty-aware label refinement on hypergraphs for personalized federated facial expression recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 35, no. 5, pp. 4675–4685, May 2025.

[6] E. Malan, C. D. Vizia, M. Castangia, V. Peluso, A. Calimera, and E. Macii, "Privacy-preserving federated learning for household characteristic identification," in *Proc. IEEE 49th Annu. Comput., Softw., Appl. Conf. (COMPSAC)*, Jul. 2025, pp. 2040–2045.

[7] T. Yang, X. Feng, S. Cai, Y. Niu, and H. Pen, "A privacy-preserving federated reinforcement learning method for multiple virtual power plants scheduling," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 72, no. 4, pp. 1939–1950, Apr. 2025.

[8] M. Bornstein, N. Nazir, J. Drgona, S. Kundu, and V. Adetola, "Finding MIDDLE ground: Scalable and secure distributed learning," in *Proc. Conf. Inf. Knowl. Manage. (CIKM)*, Oct. 2024, pp. 141–151.

[9] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, Apr. 2017, pp. 1273–1282.

[10] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, 2021.

[11] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 1–24, Jan. 2022.

[12] E. Malan, V. Peluso, A. Calimera, and E. Macii, "Adaptive client participation in budget-constrained federated learning with extreme label skew," in *Proc. 23rd IEEE Interregional NEWCAS Conf. (NEWCAS)*, Jun. 2025, pp. 15–19.

[13] X. Li and D. Zhan, "Fedrs: Federated learning with restricted softmax for label distribution non-iid data," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD)*, Aug. 2021, pp. 995–1005.

[14] J. Zhang, Z. Li, B. Li, J. Xu, S. Wu, S. Ding, and C. Wu, "Federated learning with label distribution skew via logits calibration," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jul. 2022, pp. 26 311–26 329.

[15] K. Guo, Y. Ding, J. Liang, Z. Wang, R. He, and T. Tan, "Exploring vacant classes in label-skewed federated learning," in *Proc. Conf. Artif. Intell. (AAAI)*, Feb. 2025, pp. 16 960–16 968.

[16] Q. Li, X. Li, L. Zhou, and X. Yan, "Adafl: Adaptive client selection and dynamic contribution evaluation for efficient federated learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process (ICASSP)*, Apr. 2024, pp. 6645–6649.

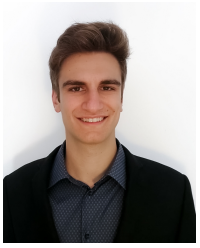
[17] L. Grativol, M. Léonardon, G. Muller, V. Fresse, and M. Arzel, "Federated learning compression designed for lightweight communications," in *Proc. IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, Dec. 2023, pp. 1–4.

[18] R. Du, D. He, Z. Ding, M. Wang, S. Chan, and X. Li, "GSASG: global sparsification with adaptive aggregated stochastic gradients for communication-efficient federated learning," *IEEE Internet Things J.*, vol. 11, no. 17, pp. 28 253–28 266, Sept. 2024.

[19] A. Herzog, R. Southam, O. Belarbi, S. Anwar, M. Bullo, P. Carnelli, and A. Khan, "Selective updates and adaptive masking for communication-efficient federated learning," *IEEE Trans. Green Commun. Netw.*, vol. 8, no. 2, pp. 852–864, Jun. 2024.

[20] J. Xu, W. Du, Y. Jin, W. He, and R. Cheng, "Ternary compression for communication-efficient federated learning," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 33, no. 3, pp. 1162–1176, Mar. 2022.

[21] H. Chen and H. Vikalo, "Mixed-precision quantization for federated learning on resource-constrained heterogeneous devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2024, pp. 6138–6148.
[22] 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. Mobile Comput.*, vol. 22, no. 12, pp. 7451–7465, 2023.
[23] Z. Li, Y. Li, B. Lin, Z. Jin, and W. Zhang, "Low precision local training is enough for federated learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2024, pp. 90 160–90 197.
[24] E. Malan, V. Peluso, A. Calimera, and E. Macii, "Communication-efficient federated learning with gradual layer freezing," *IEEE Embed. Syst. Lett.*, vol. 15, no. 1, pp. 25–28, Mar. 2023.
[25] E. Malan, V. Peluso, A. Calimera, E. Macii, and P. Montuschi, "Automatic layer freezing for communication efficiency in cross-device federated learning," *IEEE Internet Things J.*, vol. 11, no. 4, pp. 6072–6083, Feb. 2024.
[26] V. Peluso, E. Malan, A. Calimera, and E. Macii, "Private tensor freezing for an efficient federated learning with homomorphic encryption," in *Proc. IEEE Int. Conf. Comput. Des. (ICCD)*, Nov. 2024, pp. 308–315.
[27] C. Chen, H. Xu, W. Wang, B. Li, B. Li, L. Chen, and G. Zhang, "Synchronize only the immature parameters: Communication-efficient federated learning by freezing parameters adaptively," *IEEE Trans. Parallel Distributed Syst.*, vol. 35, no. 7, pp. 1155–1173, Jul. 2024.
[28] Z. Fan, R. Zhang, J. Yao, B. Han, Y. Zhang, and Y. Wang, "Federated learning with bilateral curation for partially class-disjoint data," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2023, pp. 32 006–32 019.
[29] Y. Shi, J. Liang, W. Zhang, C. Xue, V. Y. F. Tan, and S. Bai, "Understanding and mitigating dimensional collapse in federated learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 5, pp. 2936–2949, May 2024.
[30] H. Wang, P. Zheng, X. Han, W. Xu, R. Li, and T. Zhang, "Fednlr: Federated learning with neuron-wise learning rates," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD)*, Aug. 2024, pp. 3069–3080.
[31] J. Xu, M. Yang, W. Ding, and S. Huang, "Stabilizing and improving federated learning with highly non-iid data and client dropout," *Appl. Intell.*, vol. 55, no. 3, p. 216, Jan. 2025.
[32] G. Lee, M. Jeong, Y. Shin, S. Bae, and S. Yun, "Preservation of the global knowledge by not-true distillation in federated learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Nov. 2022, pp. 38 461–38 474.
[33] J. Lu, S. Li, K. Bao, P. Wang, Z. Qian, and S. Ge, "Federated learning with label-masking distillation," in *Proc. ACM Int. Conf. Multimed. (MM)*, Oct. 2023, pp. 222–232.
[34] H. Xu, J. Li, W. Wu, and H. Ren, "Federated learning with sample-level client drift mitigation," in *Proc. Conf. Artif. Intell. (AAAI)*, Apr. 2025, pp. 21 752–21 760.
[35] Y. Diao, Q. Li, and B. He, "Exploiting label skews in federated learning with model concatenation," in *Proc. Conf. Artif. Intell. (AAAI)*, Feb. 2024, pp. 11 784–11 792.
[36] H. Wang, H. Xu, Y. Li, Y. Xu, R. Li, and T. Zhang, "Fedcda: Federated learning with cross-rounds divergence-aware aggregation," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2024, pp. 1–12.
[37] S. A. Tijani, X. Ma, R. Zhang, F. Jiang, and R. Doss, "Federated learning with extreme label skew: A data extension approach," in *Proc. Int. Jt. Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–8.
[38] K. Sang, T. Rabbani, and F. Huang, "Balancing label imbalance in federated environments using only mixup and artificially-labeled noise," in *Proc. Int. Conf. Pattern Recognit. Artif. Intell. (ICPRAI)*, Jul. 2024, pp. 466–475.
[39] Y. Ma, W. Dai, W. Huang, and J. Chen, "Geometric knowledge-guided localized global distribution alignment for federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2025, pp. 20 958–20 968.
[40] Q. Ketu, W. Yi, W. Baoquan, L. Zhengdong, and Z. Xi, "Fedse: Group-based sequential training strategies for mitigating label skew in federated learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process (ICASSP)*, Apr. 2025, pp. 1–5.
[41] N. N. Hung, T. T. Nguyen, T. N. Hoang, H. H. Pham, T. H. Nguyen, and N. P. Le, "Safa: Handling sparse and scarce data in federated learning with accumulative learning," *IEEE Trans. Comput.*, vol. 74, no. 6, pp. 1844–1856, Jun. 2025.
[42] E. Malan, V. Peluso, A. Calimera, and E. Macii, "Draft & refine: Efficient resource management in federated learning under pathological labels skew," in *Proc. IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, Nov. 2024, pp. 1–4.
[43] X. Qiu, T. Parcollet, J. Fernández-Marqués, P. P. B. de Gusmao, Y. Gao, D. J. Beutel, T. Topal, A. Mathur, and N. D. Lane, "A first look into the carbon footprint of federated learning," *J. Mach. Learn. Res.*, vol. 24, no. 1, pp. 5899–5921, Jan. 2023.
[44] D. Jhunjhunwala, P. Sharma, A. Nagarkatti, and G. Joshi, "Fedvarp: Tackling the variance due to partial client participation in federated learning," in *Proc. Uncertainty Artif. Intell. (UAI)*, Aug. 2022, pp. 906–916.
[45] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Technical Report TR-2009, 2009.
[46] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2021.
[47] C. Jia, M. Hu, Z. Chen, Y. Yang, X. Xie, Y. Liu, and M. Chen, "Adaptivefl: Adaptive heterogeneous federated learning for resource-constrained aiot systems," in *Proc. Des. Autom. Conf. (DAC)*, Nov. 2024, pp. 1–6.



Erich Malan (Graduate Student Member, IEEE) received the M.Sc. degree in Data Science and Engineering from the Politecnico di Torino in 2021. He is currently pursuing the Ph.D. degree with the Department of Control and Computer Engineering, Politecnico di Torino. His main research interest focuses on distributed artificial intelligence.



Valentino Peluso (Member, IEEE) received the M.Sc. degree in electronic engineering and the Ph.D. degree in computer engineering from the Politecnico di Torino. He is currently an Assistant Professor of Computer Engineering with the Politecnico di Torino. His research interest focuses on design automation of digital circuits, dynamic power management for low-power embedded systems, edge artificial intelligence, distributed and federated learning in IoT networks.



Andrea Calimera (Member, IEEE) received the M.Sc. degree in Electronic Engineering and the Ph.D. degree in Computer Engineering from Politecnico di Torino, where he is currently a Full Professor of Computer Engineering. His research interests include design automation of digital circuits and embedded systems, with a focus on synthesis and optimization techniques for low power consumption, energy efficiency, and reliability.



Enrico Macii (Fellow, IEEE) received the Laurea degree in electrical engineering from the Politecnico di Torino, Turin, Italy, the Laurea degree in computer science from the Università di Torino, Turin, and the Ph.D. degree in computer engineering from the Politecnico di Torino received respectively in 1990, 1991, and 1995. He is currently a Full Professor of Computer Engineering with the Politecnico di Torino. His research interests include the design of electronic digital circuits and systems, with a particular emphasis on low-power consumption aspects.