Engineering Applications of Artificial Intelligence 172 (2026) 114369

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Research Paper

MAFSA: A multi-layer asynchronous federated learning with staleness-awareness in edge computing

Shiwen Zhang [a,b], Shuang Chen [a,b], Yujia Zhang [a,b], Wei Liang [a,b], Kuanching Li [a,b,*], Ling-Huey Li [c], Keqin Li [d]

[a] School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan, China
[b] Hunan Engineering Research Center of Software Design and Services of Intelligent Connected Vehicles, China
[c] Dipartimento di Informatica, Università degli Studi di Salerno, Italy
[d] Department of Computer, State University of New York, New Paltz, NY, USA

ARTICLE INFO

Keywords:
Federated learning
Edge computing
Staleness-awareness
Client dropout
Synchronous and asynchronous

ABSTRACT

Federated Learning (FL) has emerged as a promising privacy-preserving scheme in edge computing. However, traditional cloud-based FL architectures still suffer from high communication overhead, which motivates the development of hierarchical and asynchronous variants to improve communication efficiency. However, traditional cloud-side-end architecture must wait for the results of all devices to complete the update, resulting in inefficient training. Furthermore, the phenomenon of end-device dropouts can lead to the waste of resources, thereby compromising the system's fairness. In this work, we propose MAFSA, a multi-layer asynchronous federated learning scheme with staleness-awareness, which aims to enhance the system's efficiency and resource utilization while maintaining accuracy and ensuring fairness. MAFSA proposes a composite asynchronous aggregation strategy to address the inefficiency issue caused by device heterogeneity and enhance the system's communication efficiency. In addition, MAFSA proposes a staleness-awareness mechanism to cope with end device dropping, improve resource utilization, and ensure the system's fairness. Extensive experiments demonstrate that the proposed scheme effectively utilizes stale information, significantly benefiting the federated learning system and proving the method's effectiveness.

1. Introduction

In recent years, the widespread application and development of Artificial Intelligence (AI) and Machine Learning (ML) have garnered significant attention (Li et al., 2018; Cai et al., 2023; Liang et al., 2025). Traditional machine learning approaches (Bonawitz et al., 2019; Kairouz et al., 2021) typically rely on centralized data storage and processing, which, while effective, often result in significant privacy concerns and data leakage risks (Liang et al., 2023; Zhang et al., 2024, 2023). To mitigate these issues, Federated Learning (FL) has emerged as a promising paradigm, enabling collaborative model training across distributed devices without the need for data sharing (McMahan et al., 2017; Li et al., 2020a; Zhang et al., 2025). However, traditional FL frameworks assume uniform data distribution, model architecture, and device capabilities, which often diverge in real-world scenarios. This discrepancy leads to the "tail-dragging" problem, where heterogeneous device performance significantly prolongs training times (Verma et al., 2018; Yang et al., 2019). With the introduction of Mobile Edge Computing (MEC) (Lim et al., 2020; Zhao et al., 2022; Zeng et al., 2022), the application of Hierarchical Federated Learning (HFL) can leverage the rich computational and storage resources at the edge layer, thereby significantly reducing the need for expensive communication with the cloud. Comparison of traditional federated learning and client–edge–cloud hierarchical federated learning is shown in Fig. 1

Despite these advancements, the cloud-side-end FL architecture (Chai et al., 2021) faces challenges due to the limited and heterogeneous resources of end devices. Issues such as limited communication bandwidth, computational constraints, and device dropouts (Wang et al., 2020; Che et al., 2023) exacerbate inefficiencies in traditional FL frameworks, which synchronize updates only after all devices complete their training. This approach overlooks resource heterogeneity, resulting in wasted training time and reduced system efficiency. Furthermore, device dropouts during training result in underutilized resources and compromised fairness.

* Corresponding author.
E-mail addresses: shiwenzhang@hnu.edu.cn (S. Zhang), shuangchen@mail.hnust.edu.cn (S. Chen), yujiazhang@mail.hnust.edu.cn (Y. Zhang), wliang@hnust.edu.cn (W. Liang), aliric@hnust.edu.cn (K. Li), lli@unisa.it (L.-H. Li), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.engappai.2026.114369
Received 26 March 2025; Received in revised form 15 February 2026; Accepted 1 March 2026

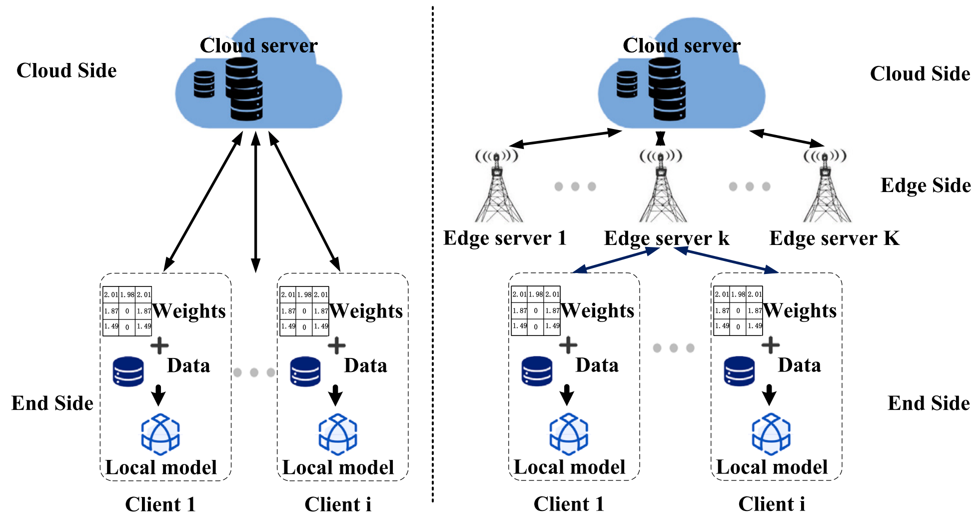


Fig. 1. Traditional federated learning vs. Client–edge–cloud hierarchical federated learning.

In real-world federated learning deployments, device heterogeneity—in terms of computation capacity, communication bandwidth, and energy availability—often leads to straggler effects and inefficient training. To directly address this challenge, several recent works (Li et al., 2020b; Yao and Sun, 2020; Karimireddy et al., 2020; Sprague et al., 2019; Yan et al., 2023; Diao et al., 2021; Liu et al., 2022; Lee et al., 2024) have proposed adaptive solutions tailored for resource-constrained clients. Some approaches (Li et al., 2020b; Yao and Sun, 2020; Karimireddy et al., 2020) enhance model generalization by incorporating regularization terms into local loss functions, mitigating the effects of data heterogeneity. However, these methods yield only a unified global model, limiting their applicability. Others (Sprague et al., 2019; Yan et al., 2023) have explored asynchronous FL to alleviate the impact of device heterogeneity, but these methods risk destabilizing global model convergence due to the aggregation of single updates. While asynchronous federated learning has been widely explored to mitigate the synchronization bottleneck caused by client heterogeneity, many existing approaches suffer from instability due to the aggregation of stale updates. Nevertheless, some studies (Diao et al., 2021; Liu et al., 2022; Lee et al., 2024) have shown that asynchronous methods can still guarantee convergence if staleness is bounded correctly. These insights inspire the design of our proposed framework, which leverages a compound-step asynchronous aggregation strategy and a staleness-aware update mechanism. Unlike prior works that discard stale updates or introduce complex regularization, our proposed framework explicitly reuses stale models with adaptive weight scaling, aiming to improve both efficiency and fairness in a hierarchical FL setup.

To address client dropout, several strategies have been proposed. Wu et al. (2021b,a) introduced client selection mechanisms to prioritize stable devices, reducing dropout rates. Nishio and Yonetani (2019) and Nunes et al. (2024) employed cutoff times to aggregate updates within a limited timeframe, improving training efficiency. However, these methods fail to fully resolve the issue, as edge devices remain resource-constrained and prone to unpredictable dropouts, resulting in underutilization of resources. In contrast to simply excluding dropped clients, our method seeks to reutilize their stale updates through a staleness-aware weighting mechanism, thus enhancing resource utilization and fairness.

To utilize resources more efficiently and increase their utilization, several strategies have been proposed. Chen et al. (2021) introduced staleness-aware asynchronous stochastic gradient descent (Async-SGD), which dynamically adjusts learning rates to mitigate the negative impact of stale updates. Xie et al. (2020) proposed FedAsync, a framework that incorporates staleness-aware aggregation mechanisms to control the influence of outdated models and ensure convergence. Luo et al. (2022) developed an adaptive client sampling strategy to tackle system and statistical heterogeneity, reducing staleness by dynamically selecting clients with timely updates. However, these methods still face challenges in fully resolving staleness, as edge devices often operate under resource constraints and unpredictable network conditions, leading to inconsistent participation and delayed updates. Further research is needed to design more robust and efficient staleness-aware algorithms that can adapt to the dynamic and heterogeneous nature of federated learning environments. Inspired by these challenges, our approach adopts a composite asynchronous aggregation strategy that maintains model stability while enhancing communication efficiency. Additionally, we incorporate a decay-based staleness adjustment scheme within our hierarchical framework to effectively control the influence of stale updates and support convergence.

In this work, we propose MAFSA, a multi-layer asynchronous federated learning scheme with staleness awareness, addressing the client dropout problem without sacrificing accuracy or efficiency. Considering network conditions, we incorporate an edge layer into the cloud architecture and adopt a composite asynchronous aggregation strategy to improve communication efficiency and accuracy. To address the client dropout problem, we subsequently propose a staleness-aware mechanism. A scaling rule for aggregated weights is utilized to simultaneously handle the client's stale model for restoring the online state and the client's fresh model for maintaining the online state, thereby reducing data loss and improving utilization. Finally, we designed a large number of experiments to prove the effectiveness of the MAFSA scheme. Overall, by addressing the limitations of existing FL frameworks, MAFSA represents a significant step forward in enabling efficient, accurate, and fair federated learning in heterogeneous edge environments. Our contributions are as follows:

- To improve the system's efficiency and resource utilization without compromising its accuracy and ensuring fairness, we propose the MAFSA scheme, a multi-layer asynchronous federated learning scheme with staleness awareness.
- To accelerate federated learning, we propose a composite asynchronous aggregation strategy. A synchronous strategy is employed within the edge server cluster, while an asynchronous strategy is utilized on the cloud servers outside the cluster to enhance efficiency.
- To address the client dropout problem, we propose a staleness-aware mechanism to improve resource usability. By scaling the weight of the expired updates before the client updates, we utilize the resources of the dropped clients and improve resource availability, ensuring the fairness of joint learning.

- To evaluate the advantages of the proposed algorithm, we conduct extensive experiments in a simulated environment. Experimental results demonstrate that our proposed algorithm fully utilizes offline clients, significantly benefits federated learning systems, and proves the method's effectiveness.

The remainder of this article is organized as follows. Related work and background are presented in Section 2. The problem statement and details of the system model are in Section 3. The proposed algorithm is depicted in Section 4, the performance of the proposed algorithm is evaluated in Section 5, and finally, concluding remarks and future directions are given in Section 6.

2. Related work

Federated Learning with inefficiency: To overcome the inefficient training on cloud servers resulting from federated learning, hierarchical federated learning has attracted significant attention. Liu et al. (2020) constructed a basic cloud–edge-end architecture that reduces the communication overhead and increases efficiency by decreasing end device model training time and energy consumption. Based on this, client heterogeneity was further considered by Luo et al. (2020), Abad et al. (2020), and Wang et al. (2021). Luo et al. (2020) designed an efficient resource scheduling algorithm to minimize the global cost through joint computation and communication resource allocation, addressing edge correlation problems. Abad et al. (2020) reduced communication delay in learning iterations by optimizing resource allocation among end devices. However, using the gradient sparsification technique brings additional computational costs and reduces the system's performance. Wang et al. (2021) employed fog computing and proposed a network-aware distributed learning approach to enable the machine learning task to scale machine learning tasks to massive datasets. This approach enhances resource utilization without compromising model accuracy when targeting data distribution across various devices. However, the end devices in this work share local data processing, raising privacy concerns in federated learning.

Federated Learning with Client Dropout: The original FedAvg algorithm (McMahan et al., 2017) performs federated learning training in an ideal environment. In actual federated learning training, client performance is highly heterogeneous with limited communication, and dropout conditions may occur at any time. Xie et al. (2020) and Sprague et al. (2019) considered device heterogeneity and employed an asynchronous federated optimization scheme, allowing devices to join midway through the training process. However, the asynchronous scheme causes the cloud server to receive too many updates, leading to server paralysis and negatively impacting the system's performance. Wu et al. (2021b) improved the asynchronous scheme and proposed a semi-asynchronous federated learning protocol. This protocol utilizes an improved pacing manipulation mechanism and model caching strategy to improve efficiency and mitigate the impact of dropped clients. Chen et al. (2017) evaluated the pros and cons of synchronous and asynchronous schemes and proposed a method to reduce lag in synchronous stochastic optimization. They introduced backup workers in synchronous optimization, which avoids asynchronous noise and reduces server waiting time during synchronization. Considering the impact on network traffic, Nishio and Yonetani (2019) selected clients based on their resource conditions and shortened the round length by filtering slow clients, thereby enabling the server to aggregate as many client updates as possible. However, it ignores client unreliability.

Federated Learning with Staleness Awareness: Staleness in federated learning, caused by asynchronous updates or client dropout, can significantly degrade the performance and convergence of the global model. Recent studies have proposed various staleness-aware mechanisms to address this challenge. Ma et al. (2024) introduced a dynamic staleness control framework for asynchronous federated learning in decentralized topologies, which adaptively adjusts the staleness threshold based on network conditions and client participation. This approach effectively balances the trade-off between model accuracy and training efficiency. Qiao et al. (2024) proposed ASMAFL, an adaptive staleness-aware momentum asynchronous federated learning algorithm, which incorporates momentum-based updates to mitigate the impact of stale gradients in edge computing environments. Similarly, Sun et al. (2024) designed a staleness-controlled asynchronous federated learning framework that dynamically adjusts the aggregation weights of stale updates to optimize the accuracy-efficiency trade-off. Chen et al. (2021) developed FedSA, a staleness-aware asynchronous federated learning algorithm tailored for non-IID data, which employs a staleness-aware aggregation strategy to improve convergence under heterogeneous data distributions. Earlier work by Xie et al. (2020) laid the foundation for asynchronous federated optimization, highlighting the challenges of staleness and proposing techniques to handle delayed updates. Additionally, Luo et al. (2022) addressed system and statistical heterogeneity in federated learning by proposing an adaptive client sampling strategy, which indirectly mitigates staleness by prioritizing reliable and timely clients. These studies collectively advance the state-of-the-art in staleness-aware federated learning, offering solutions to improve robustness and convergence in dynamic and heterogeneous environments.

3. Preliminaries

In this section, we first introduce the training process of the classical federated learning method and formally define the concept of freshness in federated learning. Subsequently, we introduce MAFSA and provide a detailed definition of the problem.

3.1. Classical federated learning

The most commonly used method in FL is Fedavg, which consists of three steps: (i) in each communication round, the server randomly selects clients to participate in training and broadcasts the latest global model to the clients selected to participate in training; (ii) the selected clients use their local datasets for training, update the model, and send the updated model to the server; and (iii) upon receiving the updated models from all participating clients, the server aggregates them to generate a new global model. The above steps are repeated until the model converges or reaches a predetermined number of communication rounds.

The goal of Fedavg is to minimize the loss function $F(\theta)$, as follows:

$$F(\theta) = \frac{1}{D} \sum_{i \in N} F_i(\theta), \tag{1}$$

where θ is the global model, N is the number of clients, $D = \sum_{i=1}^{N} |D_i|$ is the total data volume, D_i is the local dataset of client i, and $f_i(\theta) = \mathcal{L}(x_i, y_i, \theta)$ is the loss function of client i. $f_i(\theta)$ represents the loss predicted for the instance (x_i, y_i) using the model parameters. Thus, we can rewrite the objective as:

$$\theta^* = argmin f(\theta). \tag{2}$$

Unfortunately, it is unrealistic to solve this problem directly. Thereafter, the client performs a gradient descent algorithm to reach the optimization objective gradually in each training update round, as follows:

$$\theta_t^k = \theta_{t-1} - \eta \nabla f(\theta_{t-1}), \tag{3}$$

where, $\eta > 0$, η is the learning rate (step size). After the client sends the updated model to the server, the server aggregates the received models as follows:

$$\theta_{t+1} = \sum_{i=1}^{N} \frac{D_i}{D} \theta_t^k. \tag{4}$$

It is listed in Table 1 all the necessary notations used in this work.

Table 1
Summary of main notations.

Notation	Explanation
N	Number of clients
K	Number of edge servers
P_k	The set of clients for edge server k
θ	The global model
θ_k^t	Model of edge server k at round t
w_k^t	Model of client i at round t
D	Total amount of data
D_i	data of client i
x_i	Sample
y_i	Label
η	Learning rate
p_i	offline rate
t	Current training rounds
t_i	Client i' s most recent dropped round
E_{t_i}	Number of rounds dropped
α	Boosting factor
E_{avg}	Average of stale values received by the ES from all its clients
T	Total training rounds
T_c	Number of client local updates
T_e	Number of edge aggregations

3.2. Freshness

In 2011, Aluman et al. introduced the concept of information obsolescence. Later in the same year, Kaul et al. proposed the concept of the age of information. While traditional data transmission emphasizes latency, real-time update applications prioritize the timeliness of information. Federated learning systems focus on client dropout to prevent model utility loss. As data ages over time, its degree of obsolescence increases. To balance the benefits of fresh information against the costs of collecting it, we need to quantify information freshness and analyze it from a technical perspective. The freshness of information is crucial for real-time monitoring and status update systems in the Internet of Things. If the destination receives outdated information, it may compromise the accuracy and reliability of the system's decision-making, leading to significant safety risks. A measurement criterion for determining the freshness of information is its age. Suppose the time from the collection of information is defined as T_{col}, the information age of this piece of information is defined as:

$$\Delta = T_{col}, \tag{5}$$

and the information age of instantaneous information is defined as:

$$\Delta = T_{col} - g(t), \tag{6}$$

where $g(t)$ is the generation time of the latest packet received at the current receiving end, we define information freshness as below.

Definition 3.1 (*Information Freshness*)**.** The freshness of information in a system can be characterized in terms of the average age or average age distribution:

$$F = \overline{\Delta} = \frac{1}{T_{col}} \int_0^{T_{col}} \Delta(t) dt. \tag{7}$$

According to this definition, the freshness of a client's information is a measure of the expiration date of the client's previous updates. Even if the client is offline, its updates are still in progress, and this "outdated" information is crucial for system updates. Therefore, it is essential to find a solution for compelling freshness valuation.

3.3. System model

As illustrated in Fig. 2, the system model of MAFSA is divided into three parts: global aggregation, edge aggregation, and local update. Additionally, it consists of two algorithms: the staleness-awareness algorithm and the compound-step synchronous and asynchronous algorithm.

The system's training process consists of multiple rounds of global training. The first round includes all seven steps, while subsequent rounds exclude the first initialization step, solely including the second to sixth steps. The seven steps are depicted as follows:

1. **Initialization:** The cloud server initializes the global model.
2. **Global Broadcast:** The cloud server broadcasts the global model to the edge server.
3. **Edge Broadcast:** The edge server sends the cluster model to its clients.
4. **Local Training:** Online clients update the received cluster model using local data.
5. **Upload:** Clients participating in training upload the updated model and other parameters to the edge server.
6. **Edge Aggregation (Synchronous):** The edge server receives the models uploaded by all clients participating in the training within the cluster and aggregates and updates them.
7. **Edge Upload:** After several training rounds, the cluster models are uploaded to the cloud server.
8. **Global Aggregation (Asynchronous):** The cloud server performs aggregation updates for each cluster model it receives.

To illustrate the training process of MAFSA, an example is presented in Fig. 3. The system consists of one cloud server, four edge servers (numbered 1–4), and twelve clients (numbered 1–12). Clients 1–3 belong to clusters within the range of ES_1, clients 4–6 belong to clusters within the range of ES_2, clients 7–9 belong to clusters within the range of ES_3, and clients 10–12 belong to clusters within the range of ES_4, whereas each progress bar represents the time spent on the corresponding operation. The clients' performance within the range of each edge server varies, and they perform the training process independently. After local training updates, clients send the model parameters to the edge server. The edge server then selects the clients after receiving the model parameters from all available clients participating in the training within the cluster and performs the synchronization update.

In Fig. 3, clients 2, 5, and 9 drop out during the training process and do not participate in the aggregated update of the edge server to which they belong in that round. When the dropped clients return to normal, they continue participating in the training. After completing two rounds of edge aggregation, the edge server sends the model parameters to the cloud server, which updates immediately upon receiving them without waiting for other edge servers.

3.4. Problem formulation

We consider a federated learning setup consisting of a central cloud server, K edge servers, and N clients that collaboratively train a global model. We assume that each edge server authorizes the same number of clients with equal training data and that a known clustering structure exists among the users. The terms "local", "cluster", and "global" are used to denote variables, functions, and values associated with individual clients, different clusters, and the system as a whole. Each client i stores a private dataset D_i, defined as: $D_i = x_i, y_i$, where x_i is the sample and y_i is the label. And $F_i(\theta)$ is a local objective function of client i, $\mathcal{L}(\theta; x, y)$ is the loss function. In this paper, our goal is to minimize the global loss function after T iterations.

$$\begin{aligned} & minF(\theta), F(\theta) = \frac{1}{D} \sum_{i \in N} F_i(\theta), \\ & F_i(\theta) = \mathbb{E}_{(x_i, y_i) \sim D_i}[\mathcal{L}(\theta; x, y)], \\ & s.t. \begin{cases} \sum_{k=1}^{K} p_t^k = 1 & \forall t \\ \forall w_t^{k,i} = 1 & \forall k, i \\ \forall t_{k,i} < t_{bud} & \forall k, i \\ p_t^k \in (0, 1) \end{cases} \end{aligned} \tag{8}$$

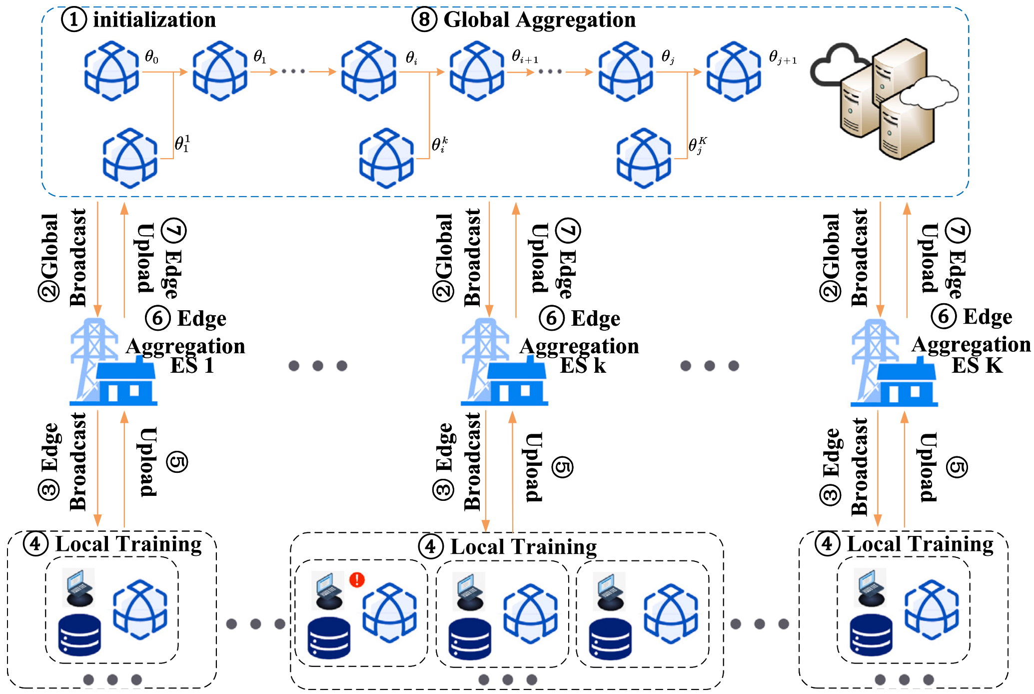


Fig. 2. System model of MAFSA.

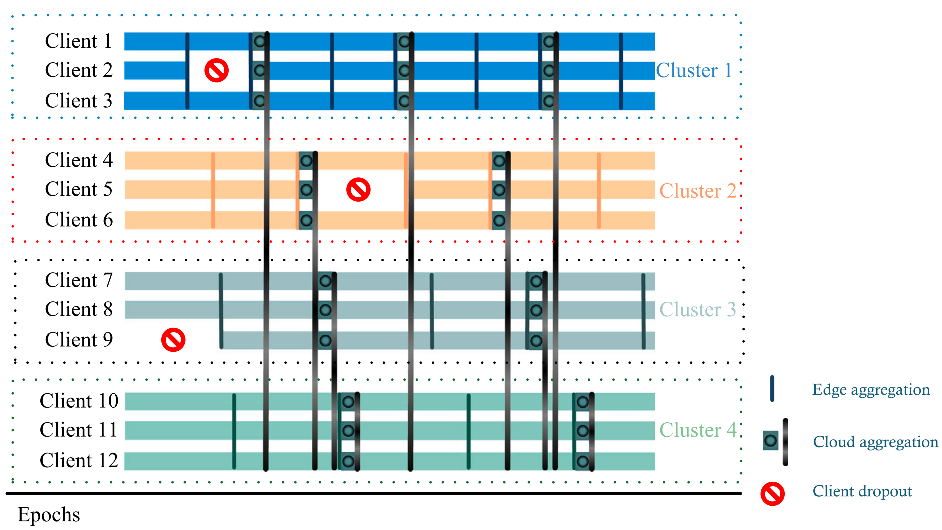


Fig. 3. Example of MAFSA training process with one cloud server, four edge servers, and twelve clients. Each horizontal bar represents the duration of local training or communication. Clients 2, 5, 9 drop out during certain rounds and later resume; the edge servers aggregate only after receiving updates from all online clients in their cluster; the cloud server aggregates asynchronously upon receiving any edge model.

Line (1): This line indicates that the edge servers participate in global updates asynchronously. That is, the cloud server aggregates the models as soon as it receives them from any edge server, without waiting for all of them.

Line (2): This line states that within each edge cluster, clients participate in edge aggregation synchronously. The edge server performs aggregation only after receiving model updates from all participating online clients in that round.

Line (3): This line imposes a time constraint: clients must complete training within a specified timeframe. Edge servers do not wait indefinitely for dropped clients; such clients are excluded from the current round of aggregation.

4. Algorithm design

To reduce communication overhead and ensure the system's fairness without compromising accuracy, we formally propose MAFSA in this section. MAFSA achieves these objectives by reducing the number of cloud aggregation rounds and employing a compound-step synchronous and asynchronous algorithm, as well as a staleness-aware algorithm.

4.1. Staleness-awareness

The ideal federated learning focuses on client-side homogeneity. However, in real-world federated learning, there is heterogeneity in network environments and hardware devices among participant clients.

These situations can lead to unstable client online statuses and waste resources as clients generate unincorporated model updates. To simulate client dropouts, we set up random sampling $\zeta_i \sim Bernoulli(p_i)$ to determine whether a client is online or not. The edge server utilizes specialized technologies to ensure that the server does not drop out randomly, thereby maintaining higher stability.

By analyzing the definition of the information age, we identify that a key challenge is measuring each client's degree of obsolescence at every update round. Also, outdated updates may be noisy and cause a client drift phenomenon. Therefore, we propose an efficient update method to address this problem and minimize its impact. This method involves multiplying the boosting factor by the stale updates and processing these stale updates together with the new updates, *i.e.*, scaling the weights of the stale updates before the client updates. The algorithm implementing this approach in a federated learning system is shown in Algorithm 1.

Algorithm 1 Staleness-Aware Algorithm

Input: T-After every T edge model aggregation, the cloud server aggregates all the edge servers' models. T_c-After every T_c local update on each client, each edge server aggregates its clients' models. p_k-The set of clients attributed to ES_K.

for all each $t = 0, \ldots, T-1$ **do**
  for all each $i = 0, \ldots, |p_k|$ **do**
    At Clients:
    Randomly sampling $\zeta_i \sim Bernoulli(p_i)$
    if $\zeta_i = 1$ **then**
      Receive θ_k^{t-1} from ES_k
      Perform T_c local update by θ_{k-1}^t
      Update expired value $E_{t_i} = t - t_i$
      Send $E_{t_i}, w_i^{t_i}, w_i^t$ to ES_k
      Update dropped rounds $t_i = 0$
    else
      if $t_i = 0$ **then**
        $t_i = t$
      end if
      Perform local update by $w_i^{t_i-1}$
    end if
  end for
  At ES_k**:**
  Receive $E_{t_i}, w_i^{t_i}, w_i^t$ from client i
  Update edge aggregation w_i^k
  $\theta_k^t = \sum_{i\in P_k} \frac{|D_i|(\alpha w_i^t + (1-\alpha) w_i^{t_i})}{\sum_{i\in P_k}|D_i|}$
  Broadcast θ_k^t to the client to which it belongs
end for

We intend to process obsolescence using the expiration value of client i. The expiration value E_{t_i} of client i is defined as the number of rounds dropped:

$$E_{t_i} = t - t_i, \tag{9}$$

where t is the number of training rounds, and t_i is the client's most recent dropped round. If a client drops out, a judgment is made on t_i. When the dropped client was online in the last round, t_i is set to the current round. When the client was also dropped, no change was made. The client that is online will send the expiration value E_{t_i}, the expired update $w_i^{t_i}$, and the latest round of update w_i^t to the edge server. Upon receiving these, the edge server performs outdated processing:

$$\theta_k^t = \sum_{i\in P_k} \frac{|D_i|(\alpha w_i^t + (1-\alpha) w_i^{t_i})}{\sum_{i\in P_k}|D_i|}, \tag{10}$$

where D_i is the data from client i, and α is a boosting factor indicating the importance of the model uploaded from client i. We then employ a function to determine the value of α:

$$\alpha(E_{t_i}) = e^{-\beta E_{t_i}}. \tag{11}$$

Then, the values of β satisfy $\frac{1}{\frac{E_{avg}}{2}+1} = e^{-\beta E_{avg}}$, *i.e.*, the inverse decay function intersects our decay function at the staleness value index. Here, E_{avg} represents the average staleness value received by the edge server from all its clients.

4.2. Composite aggregation strategy: Synchronous at edge, asynchronous at cloud

Traditional federated learning relies on a central server in the cloud; however, issues such as network congestion can result in slow communication and high latency between clients and cloud servers. In contrast, communication between edge servers and clients is more efficient. Therefore, shifting computational power from the cloud server and moving it up to the edge becomes an ideal way to improve communication efficiency, as changing the original cloud-end architecture to a cloud–edge-end architecture can significantly reduce computation delay and alleviate communication pressure.

We set up an edge server $ES_k (k \in [K])$ in each cluster. At the same time, these edge servers can be treated as homogeneous clients. In this way, a sizeable heterogeneous system can be decomposed into multiple smaller clusters. Within each cluster, a one-step synchronization method can be used to ensure robustness to data heterogeneity. A multi-step asynchronous method can be employed outside the cluster to improve communication efficiency. Thus, the heterogeneity problem can be efficiently addressed using a compound-step synchronous and asynchronous algorithm.

In this system, there is a cloud server, K edge servers with disjoint sets of clients $\{p_k\}_{k=1}^K$, and N clients. Aggregation is categorized into two types: asynchronous global aggregation (*i.e.* external aggregation) and synchronous edge aggregation (*i.e.* internal aggregation). When the cloud server receives model parameters uploaded by any edge server, it performs a global aggregation as follows:

$$\theta_{te} = \theta_{te-1} + \frac{\sum_{i\in P_k} D_i \theta_k}{\sum_{i\in[N]} D_i}. \tag{12}$$

Algorithm 2 Compound-step Synchronous and Asynchronous Algorithm

Input: T_c-After every T_c local update on each client, each edge server aggregates its clients' models. T_e: After each ES has performed T_e times of edge model aggregation, the cloud server aggregates that edge model for one global aggregation. p_k: Client sets for ES_k.

repeat
<u>Internal Synchronization</u>
At clients:
Perform T_c local updates
Update local model w_i^{tc} based on algorithm 1
Send model to $ES_k (k \in [K])$
At ES_k**:**
Receive model from client $i (i \in [p_k])$
Update edge model θ_k^{tc} based on algorithm 1
<u>External Asynchronization</u>
At ES_k**:**
if ES_k has performed T_e times of edge model aggregation **then**
  Send model θ_k to server
end if
At CS:
Receive update from ES_k
Set $te = te + 1$
Update global model θ_{te}: $\theta_{te} = \theta_{te-1} + \frac{\sum_{i\in P_k} D_i\theta_k}{\sum_{i\in[N]} D_i}$
Send model to ES_k

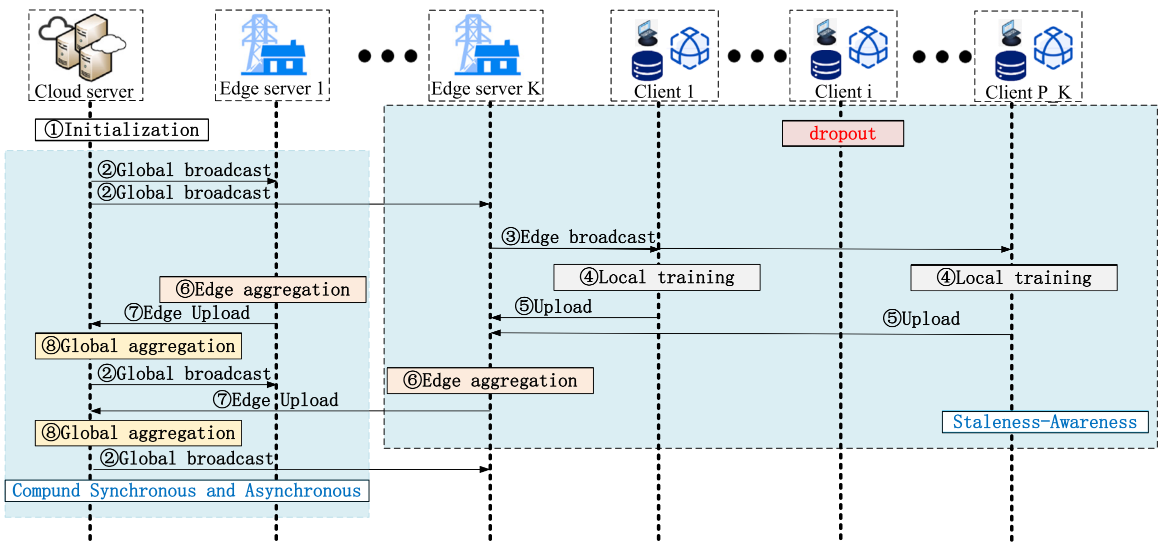


Fig. 4. Workflow of MAFSA combining staleness-aware local updates (Algorithm 1), synchronous edge aggregation, and asynchronous cloud aggregation.

An edge server performs edge aggregation only when it receives the model parameters uploaded by all online clients participating in the training.

We describe the composite asynchronous aggregation algorithm in detail in Algorithm 2. The key steps are as follows: After a sub-local update on each client, an edge server aggregates the models of its clients. When an edge server performs sub-edge aggregation, the cloud server receives the update of that edge model and performs a global aggregation.

4.3. MAFSA scheme

In this work, the MAFSA scheme is divided into two algorithms and three parts. The two algorithms are the staleness-aware algorithm and the compound-step synchronous and asynchronous algorithm, while the three parts are global aggregation, edge aggregation, and local update. In this, the first algorithm is mainly implemented in the third part and the second one primarily in the first two parts, as depicted in Fig. 4 the overall workflow of the MAFSA scheme, and Algorithm 3 a general algorithm. First is global aggregation (lines 1–4): the cloud server performs a round of global aggregation once it receives the models sent by any edge server. Next is edge aggregation (lines 5–8): the edge server performs a round of edge aggregation once it receives models uploaded by all online clients. Finally, a local update (lines 9–10): clients perform operations according to Algorithm 1.

Convergence Analysis: To provide a more detailed convergence proof for the improved federated learning algorithm, we will rigorously analyze the impact of staleness-aware updates and compound-step synchronous and asynchronous updates on the optimization process.

Assumption 4.1. Under the following assumptions:

- A1 *L-smoothness*: For all i and θ, ϕ, $\|\nabla F_i(\theta) - \nabla F_i(\phi)\| \le L\|\theta - \phi\|$
- A2 *Bounded local variance*: $\mathbb{E}_\xi\|\nabla f_i(\theta;\xi) - \nabla F_i(\theta)\|^2 \le \sigma_l^2$
- A3 *Global variance bound*: $\frac{1}{K}\sum_{i=1}^{K}\|\nabla F_i(\theta) - \nabla F(\theta)\|^2 \le \beta\|\nabla F(\theta)\|^2 + \sigma_g^2$
- A4 *Bounded staleness*: $E_{t_i} \le E_{\max} \quad \forall i, t$
- A5 *Independent client sampling*: Clients participate with probability p_i per round

---

Algorithm 3 MAFSA Scheme

Input: K edge servers $\{ES_k\}_{k=1}^{K}$, each edge server ES_k has a client set P_k, T_c: number of local updates per client, T_e: number of edge aggregations before cloud aggregation

1: **Global Aggregation at Cloud Server (CS):**
2: **if** CS receives a model from any ES_k **then**
3:   Execute **Algorithm 2** (Lines 10–19) for global asynchronous aggregation
4: **end if**
5: **Edge Aggregation at Edge Server ES_k, $\forall k \in \{1,\dots,K\}$:**
6: **if** ES_k receives models from all online clients in P_k **then**
7:   Execute **Algorithm 2** (Lines 1–9) for intra-cluster synchronous aggregation
8: **end if**
9: **Local Update at Client $i \in P_k$:**
10: Clients perform local training and staleness-awareness using **Algorithm 1**

Theorem 4.1 (*Convergence of MAFSA*). *Under Assumption 4.1, for learning rate* $\eta \le \min\left\{\frac{1}{L\sqrt{8(\beta+1)E_{\max}^2}}, \frac{1}{\sqrt{T}}\right\}$, *after T rounds, we have:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\theta_t)\|^2 \le \mathcal{O}\left(\frac{F(\theta_0) - F^*}{\eta T}\right) + \mathcal{O}\left(\eta L\sigma^2\right) + \mathcal{O}\left(\eta^2 E_{\max}^2 L^2\sigma^2\right). \tag{13}$$

where $\sigma^2 = \sigma^2 + \sigma_g^2$ and F^* is the infimum of F. Hence, with $\eta = \mathcal{O}(\frac{1}{\sqrt{T}})$, the right-hand side vanishes as $T- > \infty$, guaranteeing convergence to a stationary point.

We follow a standard non-convex SGD analysis framework extended to account for staleness-aware aggregation and hierarchical asynchronous updates.

Step 1: Descent lemma (L-smoothness, A1). By L-smoothness of each F_i, the global loss satisfies

$$\mathbb{E}[F(\theta_{t+1})|\theta_t] \le F(\theta_t) - \eta\langle\nabla F(\theta_t), \mathbb{E}[g_t|\theta_t]\rangle + \frac{\eta^2 L}{2}\mathbb{E}[\|g_t\|^2|\theta_t] \tag{14}$$

Step 2: Bias–variance decomposition. Write

$$\mathbb{E}[F(\theta_{t+1})|\theta_t] \le F(\theta_t) - \eta\langle \nabla F(\theta_t), \mathbb{E}[g_t|\theta_t]\rangle + \frac{\eta^2 L}{2}\left(B_t + V_t + \|\nabla F(\theta_t)\|^2\right) \quad (15)$$

The first term is the staleness-induced bias B_{t_i}, the second is the variance V_t.

Step 3: Bounding Staleness-Induced Bias (B_t) Using L-smoothness (A1), bounded staleness (A4), and the local update rule,

$$\|\mathbb{E}[g_t] - \nabla F(\theta_t)\| \le L\sqrt{\frac{1}{K}\sum_{i=1}^{K}\mathbb{E}\|w_i^{t_i} - \theta_t\|^2}. \quad (16)$$

The deviation $\|w_i^{t_i} - \theta_t\|^2$ is at most $\eta\mathbb{E}_{max} max_{k\in[t_i,t]}\|\nabla F_i(\theta_k)\|$, leading to

$$\mathbb{E}\|w_i^{t_i} - \theta_t\|^2 \le \eta^2 E_{\max}^2 \mathbb{E}\|\nabla F_i(\theta_t)\|^2. \quad (17)$$

Applying the global variance bound (A3) yields

$$B_t \le 2\eta^2 E_{\max}^2 L^2(1+\beta)\|\nabla F(\theta_t)\|^2 + 2\eta^2 E_{\max}^2 L^2 \sigma_g^2 \quad (18)$$

Step 4: Bounding the variance V_t**.** Local stochastic noise (A2) and independent sampling (A5) give

$$V_t = \mathbb{E} \parallel g_t - \mathbb{E}[g_t]\|^2 \le \frac{\sigma_l^2}{K} \quad (19)$$

Step 5: Combining and telescoping. Substituting the bounds of B_t and V_T into the descent inequality and applying Young's inequality, after algebraic manipulation we obtain

$$\mathbb{E}[F(\theta_{t+1})] \le \mathbb{E}[F(\theta_t)] - \frac{\eta}{4}\mathbb{E}\|\nabla F(\theta_t)\|^2 + \eta^2 L\frac{\sigma_i^2}{2k} + \mathcal{O}(\eta^3 E_{\max}^2 L^3 \sigma_g^2) \quad (20)$$

provided η is chosen sufficiently small (as in Theorem 4.1). Summing over $t = 0, 1, \ldots, T-1$ and rearranging gives the stated convergence rate.

A complete, rigorous derivation with detailed algebraic steps and auxiliary lemmas is provided in Appendix.

5. Experimental results

In this section, we conduct a series of experiments to evaluate the effectiveness of our proposed algorithm. We discuss the experimental setup in detail and then present the experimental results obtained by training on two datasets. Finally, we provide a summary and discussion of these results.

5.1. Experiment settings

Datasets and models: In this work, we use two widely recognized benchmark datasets, CIFAR-10 (Krizhevsky, 2009) and MNIST (Lecun et al., 1998), to evaluate the performance of MAFSA under different data distributions. The CIFAR-10 dataset consists of 10 classes of 60,000 color images in 32 ∗ 32 resolution, with 6000 images per class. The MNIST dataset consists of 70,000 handwritten digit images in 28 ∗ 28 resolution. To adapt to the federated learning environment, we normalized the data by scaling the pixel values to the range of $[0, 1]$, randomly cropped and flipped each dataset, and then generated the number of samples per client based on the Dirichlet distribution for Non-Independent and Identically Distributed (Non-IID) data segmentation, mimicking real-world data distribution inhomogeneity.

We used several neural network architectures for training to adapt to the characteristics of the MNIST and CIFAR-10 datasets and to evaluate their performance in a federated learning environment. For the CIFAR-10 dataset, we designed a Convolutional Neural Network (CNN) composite model consisting of a shared layer containing three convolutional layers, a specific layer containing a dropout layer, and two fully connected layers. For the MNIST dataset, we used two models, Lenet and Logistic, respectively. Lenet is a convolutional model with two convolutional layers and two fully connected layers, while Logistic is a simple logistic regression classifier.

Baseline methods: We compare the proposed approach, MAFSA, with two approaches of Hierfedavg (Liu et al., 2020), the initial Hierfedavg and Hierfedavg in the client exit state of cloud–edge–client. HierFedAvg is a baseline framework adapted from the classic FedAvg algorithm and extended to a hierarchical (cloud–edge–client) architecture. It is widely adopted in the literature as a reference implementation for multi-tier federated learning systems (Sprague et al., 2019; Yan et al., 2023; Diao et al., 2021; Liu et al., 2022; Chen et al., 2021; Liu et al., 2020; Luo et al., 2020; Abad et al., 2020). Although not a single formally published algorithm, it represents a standard and commonly used benchmark for evaluating hierarchical FL frameworks like ours.

Evaluation metrics: To comprehensively evaluate our approach, we employed three different metrics. We assessed the classification results trained by our scheme based on the final model accuracy and convergence speed (the time taken to reach the target accuracy). Additionally, we calculated the performance standard deviation for each testing round to determine performance stability.

Experiment parameters: The experiments simulate a hierarchical federated learning environment consisting of one cloud server, five edge servers, and 100 clients, with each edge server managing 20 clients (Liu et al., 2020; Luo et al., 2020; Abad et al., 2020). We set the probability of a client going offline to P_i. In Liu et al. (2024), we set P_i to 0.03 for some experiments. All models use the Adam optimizer with an initial learning rate lr of 0.01, a learning rate decay coefficient lr_decay of 0.995, and a learning rate decay period lr_decay_epoch of one epoch. The batch size is set to 20. The process of training communication rounds lasts for 100 rounds, with each client performing 60 local updates per round. Unless otherwise specified, the edge server performs one aggregation per communication round.

5.2. Effects of the proposed scheme

Impact of Client Dropout on Model Performance

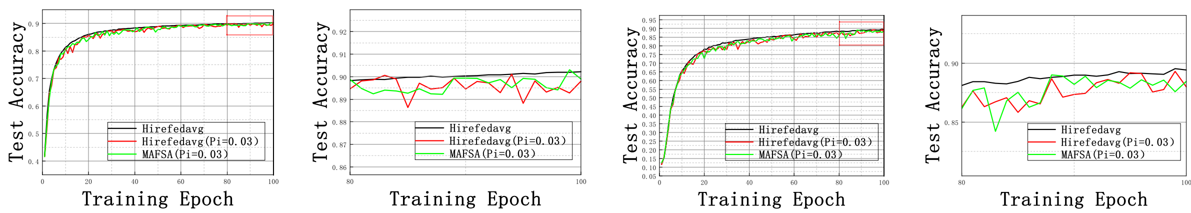
Our first set of experiments analyzes the effect of client offline events on the efficiency of federated learning training. We compare the performance of MAFSA, the baseline under ideal conditions, and the baseline under client dropout conditions, as illustrated by the performance of different convolutional neural network models on the two datasets depicted in Table 2, Figs. 5 and 6.

In Figs. 5(b), (d) and 6(b), there are small fluctuations in the accuracy curves in the later stages of training, which may be due to incomplete parameter aggregation in some rounds caused by client dropouts. Figs. 5 and 6 show that under ideal conditions, the federated learning scheme Hierfedavg is superior in terms of convergence speed, accuracy, and stability. However, in reality, due to factors such as device heterogeneity and unstable network conditions, the federated learning scheme Hierfedavg ($P_i = 0.03$) has a decrease in model accuracy and stability in the presence of dropouts. This also illustrates the need for federal learning schemes to consider the necessity of customer dropout. In contrast, our scheme takes client dropout into account.MAFSA's accuracy on Cifar-10 (0.495) is the same as Hirefedavg's, but the standard deviation (0.091) is lower, indicating that it is more robust to client dropout. Remarkably, MAFSA converges with the federated learning scheme Hierfedavg regarding accuracy in ideal conditions. It takes less time to complete, converges faster, and is more stable than the federated learning scheme Hierfedavg ($P_i = 0.03$), which suffers from dropouts.

Second, Table 2 compares the performance of the Logistic and Lenet models on the MNIST dataset and the CNN model on the CIFAR-10 dataset. Table 2 further compares the performance of the three schemes (federated learning scheme Hierfedavg in ideal conditions, federated learning scheme Hierfedavg ($P_i = 0.03$) in the presence of dropouts,

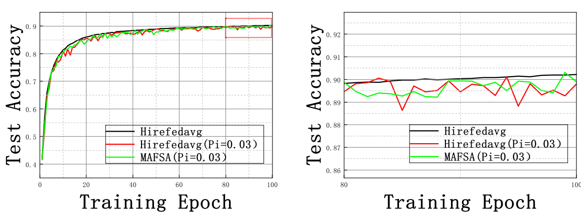
Table 2
Comparison of the performance of different schemes in training different models with different datasets.

Method	Cifar-10			MNIST					
	CNN			Lenet			Logstic		
	Accuracy	Acc-std	Time	Accuracy	Acc-std	Time	Accuracy	Acc-std	Time
Hirefedavg	0.495	0.097	13 053	0.894	0.149	6912	0.902	0.069	6217
HireFedavg(Pi = 0.03)	0.480	0.095	13 474	0.878	0.148	7164	0.898	0.072	6613
MAFSA(Pi = 0.03)	**0.495**	**0.091**	**13 438**	**0.884**	**0.146**	**7045**	**0.900**	**0.071**	**6379**



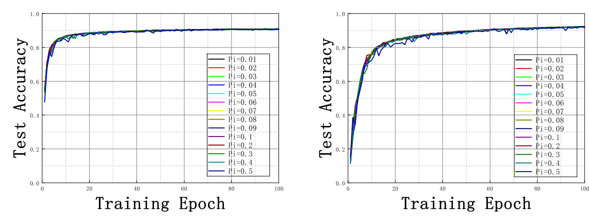
(a) Logistic-ACC (b) Logistic-ACC′ (c) Lenet-ACC (d) Lenet-ACC′

Fig. 5. Accuracy of Logistic and Lenet on MNIST versus time rounds. Where (b) is a subgraph of (a), (d) is a subgraph of (c).



(a) CNN-ACC (b) CNN-ACC′

Fig. 6. Accuracy of CNN on CIFAR-10 versus time rounds. Where (b) is a subgraph of (a).



(a) Logistic (b) Lenet

Fig. 7. Impact of different offline rates on the MAFSA scheme over MNIST. Left plot: impact of different offline rates on MAFSA scheme using Logistic models over MNIST. Right plot: impact of different offline rates on MAFSA scheme using Lenet models over MNIST.

and MAFSA (P_i = 0.03)) using data. Remarkably, MAFSA converges with the federated learning scheme Hierfedavg in terms of accuracy in ideal conditions. The accuracy of Hirefedavg on Cifar-10 in Table 2 is slightly higher (0.495) than its variant (0.480), possibly due to the latter's incomplete compensation for client dropouts in some rounds. Hirefedavg's Time is 13 053, while its variant (P_i = 0.03) is slightly higher (13 474), suggesting that client dropouts may require additional rounds or a waiting mechanism, thus lengthening the overall training time. Compared to the federated learning scheme Hierfedavg (P_i = 0.03), MAFSA has a shorter convergence time, faster convergence, and better stability.

By comparing the federated learning scheme Hierfedavg in two conditions (the ideal condition, where the federated learning scheme Hierfedavg is applied, and the federated learning scheme Hierfedavg (P_i = 0.03) in the presence of dropouts), it is illustrated that client offline events lead to problems such as resource wastage and degradation of model performance. Our proposed MAFSA accounts for client dropouts and is reluctant to give up stale model update information. Although some model performance is sacrificed, the accuracy and convergence speed performance converge to the ideal condition federated learning scheme Hierfedavg.

Effects of offline rate

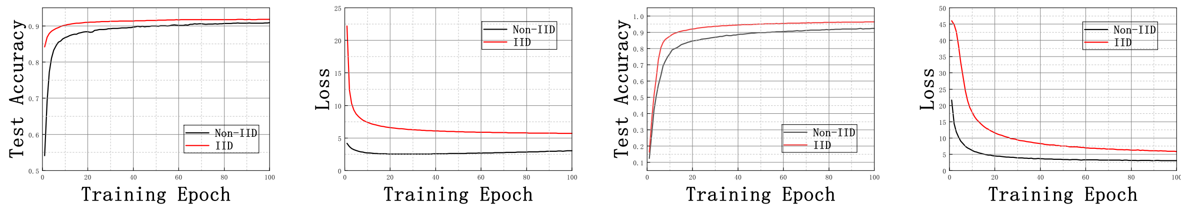
The client dropout problem in federated learning can lead to reduced communication efficiency, training delay, resource waste, convergence fluctuations, and degradation of model robustness within the system. Specifically, a high offline rate causes the server to wait or retry frequently, which significantly slows down the global convergence speed; an unstable number of clients effectively updated in each round leads to global model parameter oscillations and unsmooth convergence curves; data from frequently dropped clients (e.g., a specific group of devices) are not aggregated for a long period of time, which leads to a global model biased towards the clients that are stably online, and reduces the generalization ability.

Logistic regression has fewer parameters and may be less sensitive to changes in data volume, whereas Lenet, as a deep learning model, may be more susceptible to data reduction. We illustrate the robustness of the MAFSA scheme by analyzing the effect of different offline rates on our designed scheme MAFSA. As shown in Fig. 7(a) and (b), the MAFSA scheme achieves high accuracy at different client offline rates. We observe that at high offline rates (e.g., >20%), fewer clients participate in training, and the global model update relies on fewer local gradients, which may lead to slower convergence. The deep structure of Lenet is more sensitive to gradient noise, and the volatility of the convergence process may increase. During training, the model becomes unstable as the offline rate increases. However, the model stabilizes as the number of training rounds increases. Experiments prove that our proposed MAFSA scheme is robust.

In the MAFSA scheme, the setting of offline rate needs to be adjusted according to the model complexity. For the Logistic regression model, a high offline rate (e.g., 20%–30%) can be tolerated, while complex models such as Lenet require a lower offline rate (e.g., <15%) to maintain performance. A high offline rate may lead to an unstable training process, large fluctuations in model parameter updates, and slower convergence.

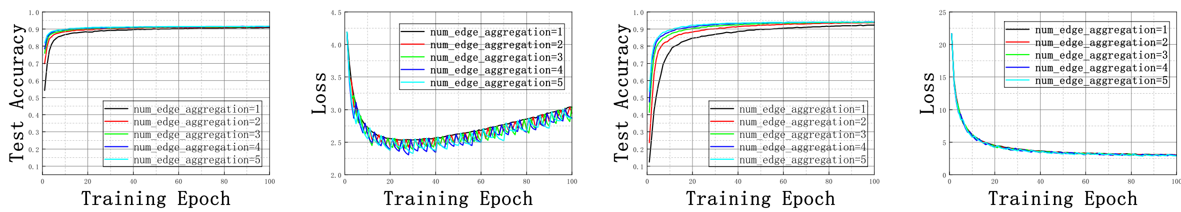
Effects of data heterogeneity

Data heterogeneity mainly affects model performance through uneven feature distribution and category imbalance. We analyze the impact of data heterogeneity on the federated learning system. Fig. 8 compares the accuracy and loss of different models of this paper's



(a) Logistic-Acc (b) Logistic-Loss (c) Lenet-Acc (d) Lenet-Loss

Fig. 8. Impact of data heterogeneity on MAFSA scheme over MNIST. (a) is the effect of data heterogeneity on test accuracy when using the Logistic model. (b) is the effect of data heterogeneity on loss when using the Logistic model. (a) is the effect of data heterogeneity on test accuracy when using the Lenet model. (a) is the effect of data heterogeneity on loss when using the Lenet model.



(a) Logistic-Acc (b) Logistic-Loss (c) Lenet-Acc (d) Lenet-Loss

Fig. 9. Impact of the number of edge aggregations on the MAFSA scheme over MNIST. (a) is the effect of the number of edge aggregations on test accuracy when using the Logistic model. (b) is the effect of the number of edge aggregations on loss when using the Logistic model. (c) is the effect of the number of edge aggregations on test accuracy when using the Lenet model. (d) is the effect of the number of edge aggregations on loss when using the Lenet model.

algorithm in Independent and Identically Distributed (IID) and Non-IID settings. As shown in Fig. 8, the model under the IID setting is more accurate, converges faster, and has lower loss than the model under the Non-IID setting for the same number of training rounds. From Fig. 8, it can be concluded that the logistic regression model and the lenet model are significantly less accurate and have more fluctuating loss values when data heterogeneity is high. This indicates that the models are sensitive to data distribution inhomogeneity, and it is difficult to extract effective features from heterogeneous data, resulting in insufficient generalization ability.

It can be concluded that even in the case of data heterogeneity, the MAFSA scheme converges faster, and the accuracy of the Logistic model can reach 90.9%, and the accuracy of the Lenet model can reach 92.35%. The MAFSA scheme can effectively improve accuracy and training stability in heterogeneous data scenarios through algorithmic optimization. The experimental results confirm that the MAFSA scheme effectively mitigates the impact of data heterogeneity.

Effects of edge aggregation rounds

More rounds of edge aggregation imply more rounds of parameter synchronization and global optimization, which can mitigate the effect of uneven data distribution among edge devices. Ultimately, we analyze the impact of the number of edge aggregation rounds on model performance, and the results are presented in Fig. 9.

Fig. 9(a) and (c) show that increasing the number of edge aggregation rounds can accelerate the convergence speed of the model and improve the model accuracy. The logistic regression model gradually increases its accuracy with an increase in the number of aggregation rounds. Lenet model's accuracy increases significantly with the increase in rounds. It is able to optimize the parameters in more rounds because it is more adapted to image feature extraction. Fig. 9(b) and (d) show that the loss of the Logistic model decreases and then increases. The loss decreases more rapidly at the initial stage, and subsequently, the loss value decreases more slowly or even stagnates due to the model's capacity limitation. The Lenet model, on the other hand, can generalize effectively, and the loss value decreases continuously with a larger magnitude, indicating that the model gradually converges to a better solution through multiple rounds of aggregation. Logistic regression, as a linear model, is limited by its expression capacity, and it is difficult to capture the complex features of the MNIST dataset, even if the number of iterations is increased, so there is a bottleneck in performance enhancement. The Lenet model is able to gradually optimize the local features through multiple rounds of aggregation due to the presence of convolutional layers and nonlinear activation functions. The LeNet model, with its convolutional layer and nonlinear activation function, is able to optimize the local and global features step by step through multiple rounds of aggregation, and its performance improves significantly with each round.

For the Lenet model, multiple rounds of aggregation can effectively improve the model's robustness; however, for the logistic regression model, too many rounds may lead to diminishing marginal gains. Therefore, selecting an appropriate model for training remains crucial for further research. The experimental results demonstrate that edge aggregation can accelerate the training process and improve the training effect.

5.3. Summary and discussions of experiment results

Overall, our proposed scheme ensures the system's fairness by reducing communication overhead and mitigating the effects of data heterogeneity without affecting the system's accuracy. The first experiment demonstrates the importance of considering offline clients in federated learning systems. The second experiment shows that our scheme is robust to client dropouts. The third experiment indicates that our scheme somewhat mitigates the effects of data heterogeneity. The final experiment highlights the importance of edge aggregation. These four experiments collectively prove the effectiveness and robustness of the MAFSA scheme.

6. Concluding remarks and future work

To address the client dropout problem without sacrificing accuracy and efficiency, we propose MAFSA, a multi-layer asynchronous federated learning scheme with staleness awareness. Unlike existing studies, our proposed scheme handled the dropout problem by reusing the knowledge from previously dropped clients, thereby avoiding resource wastage. To minimize communication overhead, we introduced mobile edge computing into the federated learning framework, transforming the traditional cloud-end architecture into a cloud–edge-end architecture. We also designed a composite asynchronous aggregation strategy, employing a synchronous policy within the edge server cluster and an asynchronous policy on the cloud side, which significantly reduced computation delay and alleviated communication pressure. Additionally, we developed a staleness-aware mechanism to enhance resource utilization and ensure fairness. By applying a weight-scaling rule to handle both stale and fresh client models, our scheme effectively mitigated the negative impact of client dropout. Experimental results demonstrated that MAFSA leveraged stale client updates and outperformed baseline methods in terms of efficiency, robustness, and convergence speed.

future work, we plan to systematically evaluate the scalability of our framework by investigating the impact of different hierarchical configurations, such as the number of edge servers and cluster sizes, on system performance. Furthermore, we will explore improved client selection strategies and examine how to integrate heterogeneous data types for broader application scenarios. Finally, we aim to extend the current framework to support more complex FL tasks with enhanced privacy and security guarantees.

CRediT authorship contribution statement

Shiwen Zhang: Validation, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization. **Shuang Chen:** Writing – original draft, Resources, Investigation, Formal analysis, Data curation. **Yujia Zhang:** Writing – original draft, Visualization, Resources, Methodology, Investigation, Formal analysis. **Wei Liang:** Writing – review & editing, Visualization, Supervision, Resources, Investigation. **Kuanching Li:** Writing – review & editing, Visualization, Validation, Project administration, Methodology. **Ling-Huey Li:** Data curation, Resources, Software, Visualization. **Keqin Li:** Writing – review & editing, Validation, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is supported in part by the Ministry of Education (China) Project of Humanities and Social Sciences (No. 25YJAZH242), the Scientific Research Fund of Hunan Provincial Education Department (No. 24A0337), and the Natural Science Foundation of Hunan Province (No. 2025JJ50348).

Appendix

Proof. We provide a complete derivation of the non-convex convergence result through five technical steps:

Step 1: Descent Lemma (Using L-smoothness) By Assumption (A1), for any θ_t and update $\theta_{t+1} = \theta_t - \eta g_t$:

$$F(\theta_{t+1}) \le F(\theta_t) + \langle \nabla F(\theta_t), \theta_{t+1} - \theta_t \rangle + \frac{L}{2}\|\theta_{t+1} - \theta_t\|^2$$
$$= F(\theta_t) - \eta\langle \nabla F(\theta_t), g_t \rangle + \frac{\eta^2 L}{2}\|g_t\|^2 \tag{21}$$

Taking expectation conditioned on θ_t:

$$\mathbb{E}[F(\theta_{t+1})|\theta_t] \le F(\theta_t) - \eta\langle \nabla F(\theta_t), \mathbb{E}[g_t|\theta_t] \rangle + \frac{\eta^2 L}{2}\mathbb{E}[\|g_t\|^2|\theta_t] \tag{22}$$

Step 2: Bias–Variance Decomposition of Gradient Define the gradient bias and variance terms:

$$\mathbb{E}[\|g_t\|^2|\theta_t] = \underbrace{\|\mathbb{E}[g_t|\theta_t] - \nabla F(\theta_t)\|^2}_{\text{Bias } B_t} + \underbrace{\mathbb{E}[\|g_t - \mathbb{E}[g_t|\theta_t]\|^2|\theta_t]}_{\text{Variance } V_t} + \|\nabla F(\theta_t)\|^2 \tag{23}$$

Substitute into (22):

$$\mathbb{E}[F(\theta_{t+1})|\theta_t] \le F(\theta_t) - \eta\langle \nabla F(\theta_t), \mathbb{E}[g_t|\theta_t] \rangle + \frac{\eta^2 L}{2}\left(B_t + V_t + \|\nabla F(\theta_t)\|^2\right) \tag{24}$$

Step 3: Bounding Staleness-Induced Bias (B_t) Let $\delta_{t,i} = w_i^{t_i} - \theta_t$ be the model deviation due to staleness. By L-smoothness (A1):

$$\|\mathbb{E}[g_t] - \nabla F(\theta_t)\| = \left\| \frac{1}{K}\sum_{i=1}^{K} \mathbb{E}\left[\nabla F_i(w_i^{t_i}) - \nabla F_i(\theta_t)\right] \right\|$$
$$\le \frac{1}{K}\sum_{i=1}^{K} \mathbb{E}\left\|\nabla F_i(w_i^{t_i}) - \nabla F_i(\theta_t)\right\|$$
$$\le \frac{L}{K}\sum_{i=1}^{K} \mathbb{E}\|w_i^{t_i} - \theta_t\| \quad \text{(by A1)}$$
$$\le L\sqrt{\frac{1}{K}\sum_{i=1}^{K} \mathbb{E}\|w_i^{t_i} - \theta_t\|^2} \quad \text{(Jensen)} \tag{25}$$

By bounded staleness (A4) and local update rule:

$$\|w_i^{t_i} - \theta_t\| \le \sum_{k=t_i}^{t-1} \eta\|\nabla F_i(\theta_k)\| \le \eta E_{\max} \max_{k\in[t_i,t]} \|\nabla F_i(\theta_k)\|$$
$$\Rightarrow \mathbb{E}\|w_i^{t_i} - \theta_t\|^2 \le \eta^2 E_{\max}^2 \mathbb{E}\|\nabla F_i(\theta_t)\|^2 \tag{26}$$

Using global variance bound (A3):

$$\frac{1}{K}\sum_{i=1}^{K} \mathbb{E}\|\nabla F_i(\theta_t)\|^2 \le \frac{1}{K}\sum_{i=1}^{K}\left(2\mathbb{E}\|\nabla F_i(\theta_t) - \nabla F(\theta_t)\|^2 + 2\|\nabla F(\theta_t)\|^2\right)$$
$$\le 2(\beta\|\nabla F(\theta_t)\|^2 + \sigma_g^2) + 2\|\nabla F(\theta_t)\|^2$$
$$= 2(1+\beta)\|\nabla F(\theta_t)\|^2 + 2\sigma_g^2 \tag{27}$$

Combining (25), (26), and (27):

$$B_t \le \eta^2 E_{\max}^2 L^2 \left[2(1+\beta)\|\nabla F(\theta_t)\|^2 + 2\sigma_g^2\right]$$
$$= 2\eta^2 E_{\max}^2 L^2 (1+\beta)\|\nabla F(\theta_t)\|^2 + 2\eta^2 E_{\max}^2 L^2 \sigma_g^2 \tag{28}$$

Step 4: Bounding Gradient Variance (V_t) Using local variance (A2) and global divergence (A3):

$$V_t = \mathbb{E}\left\| \frac{1}{K}\sum_{i=1}^{K}\left(\nabla f_i(w_i^{t_i};\xi_i) - \mathbb{E}[\nabla f_i(w_i^{t_i})]\right) \right\|^2$$
$$= \frac{1}{K^2}\sum_{i=1}^{K} \mathbb{E}\|\nabla f_i(w_i^{t_i};\xi_i) - \nabla F_i(w_i^{t_i})\|^2 \quad \text{(i.i.d. noise)}$$
$$\le \frac{1}{K^2}\sum_{i=1}^{K}\sigma_l^2 = \frac{\sigma_l^2}{K} \tag{29}$$

Additionally:

$$\mathbb{E}\|g_t - \nabla F(\theta_t)\|^2 \le 2\mathbb{E}\|g_t - \mathbb{E}[g_t]\|^2 + 2\|\mathbb{E}[g_t] - \nabla F(\theta_t)\|^2$$
$$\le \frac{2\sigma_l^2}{K} + 2B_t \tag{30}$$

Step 5: Combining Results and Summing over T Rounds Substitute (28) and (29) into (24):

$$\mathbb{E}[F(\theta_{t+1})|\theta_t] \le F(\theta_t) - \eta\|\nabla F(\theta_t)\|^2 + \eta\|\nabla F(\theta_t)\| \cdot \|\mathbb{E}[g_t] - \nabla F(\theta_t)\| + \frac{\eta^2 L}{2}\left(B_t + V_t + \|\nabla F(\theta_t)\|^2\right) \tag{31}$$

$$\leq F(\theta_t) - \eta\|\nabla F(\theta_t)\|^2 + \frac{\eta}{2}\|\nabla F(\theta_t)\|^2 + \frac{\eta}{2}B_t \quad \text{(Young's)}$$
$$+ \frac{\eta^2 L}{2}\left(B_t + \frac{\sigma_l^2}{K} + \|\nabla F(\theta_t)\|^2\right) \tag{32}$$
$$= F(\theta_t) - \left(\frac{\eta}{2} - \frac{\eta^2 L}{2}\right)\|\nabla F(\theta_t)\|^2 + \left(\frac{\eta}{2} + \frac{\eta^2 L}{2}\right)B_t$$
$$+ \frac{\eta^2 L\sigma_l^2}{2K} \tag{33}$$

Now substitute bias bound (28):

$$\mathbb{E}[F(\theta_{t+1})|\theta_t] \leq F(\theta_t) - \left(\frac{\eta}{2} - \frac{\eta^2 L}{2}\right)\|\nabla F(\theta_t)\|^2$$
$$+ \left(\frac{\eta}{2} + \frac{\eta^2 L}{2}\right)\left[2\eta^2 E_{\max}^2 L^2(1+\beta)\|\nabla F(\theta_t)\|^2\right.$$
$$\left.+2\eta^2 E_{\max}^2 L^2\sigma_g^2\right]$$
$$+ \frac{\eta^2 L\sigma_l^2}{2K} \tag{34}$$
$$= F(\theta_t) - \eta\left[\frac{1}{2} - \frac{\eta L}{2} - \eta^2 E_{\max}^2 L^2(1+\beta)(1+\eta L)\right]$$
$$\times \|\nabla F(\theta_t)\|^2$$
$$+ \eta^2\left[\frac{L\sigma_l^2}{2K} + \eta^2 E_{\max}^2 L^2(1+\eta L)\sigma_g^2\right] \tag{35}$$

For $\eta \leq \min\left\{\frac{1}{L\sqrt{8(1+\beta)E_{\max}^2}}, \frac{1}{\sqrt{T}}\right\}$, the coefficient satisfies:

$$c(\eta) := \frac{1}{2} - \frac{\eta L}{2} - \eta^2 E_{\max}^2 L^2(1+\beta)(1+\eta L) \geq \frac{1}{4}$$

Taking total expectation and summing from $t = 0$ to $T-1$:

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla F(\theta_t)\|^2 \leq \frac{4}{\eta T}\mathbb{E}[F(\theta_0) - F(\theta_T)]$$
$$+ 4\eta\left[\frac{L\sigma_l^2}{2K} + \eta^2 E_{\max}^2 L^2(1+\eta L)\sigma_g^2\right] \tag{36}$$
$$\leq \frac{4(F(\theta_0) - F^*)}{\eta T} + 2\eta L\left(\frac{\sigma_l^2}{K} + \sigma_g^2\right) + \mathcal{O}(\eta^3 E_{\max}^2 L^3\sigma_g^2) \tag{37}$$

where $\sigma^2 = \sigma_l^2 + \sigma_g^2$. This completes the proof of (13). □

Data availability

The authors do not have permission to share data.

References

Abad, M.S.H., Ozfatura, E., GUndUz, D., Ercetin, O., 2020. Hierarchical federated learning ACROSS heterogeneous cellular networks. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP, pp. 8866–8870.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečný, J., Mazzocchi, S., McMahan, B., Van Overveldt, T., Petrou, D., Ramage, D., Roselander, J., 2019. Towards federated learning at scale: System design. In: Talwalkar, A., Smith, V., Zaharia, M. (Eds.), Proceedings of Machine Learning and Systems. Vol. 1, pp. 374–388, [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2019/file/7b770da633baf74895be22a8807f1a8f-Paper.pdf.

Cai, J., Liang, W., Li, X., Li, K., Gui, Z., Khan, M.K., 2023. GTxChain: A secure IoT smart blockchain architecture based on graph neural network. IEEE Internet Things J. 10 (24), 21502–21514.

Chai, H., Leng, S., Chen, Y., Zhang, K., 2021. A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles. IEEE Trans. Intell. Transp. Syst. 22 (7), 3975–3986.

Che, T., Liu, J., Zhou, Y., Ren, J., Zhou, J., Sheng, V., Dai, H., Dou, D., 2023. Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization. In: Bouamor, H., Pino, J., Bali, K. (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Singapore, pp. 7871–7888, [Online]. Available: https://aclanthology.org/2023.emnlp-main.488.

Chen, M., Mao, B., Ma, T., 2021. FedSA: A staleness-aware asynchronous federated learning algorithm with non-IID data. Future Gener. Comput. Syst. 120, 1–12, [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X21000649.

Chen, J., Pan, X., Monga, R., Bengio, S., Jozefowicz, R., 2017. Revisiting distributed synchronous SGD. [Online]. Available: https://arxiv.org/abs/1604.00981.

Diao, E., Ding, J., Tarokh, V., 2021. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. In: International Conference on Learning Representations. ICLR.

Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al., 2021. Advances and open problems in federated learning. Found. Trends® Mach. Learn. 14 (1–2), 1–210.

Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S.J., Stich, S.U., Suresh, A.T., 2020. SCAFFOLD: Stochastic controlled averaging for federated learning. In: Proceedings of the 37th International Conference on Machine Learning. ICML '20, JMLR.org.

Krizhevsky, A., 2009. Learning multiple layers of features from tiny images. [Online]. Available: https://api.semanticscholar.org/CorpusID:18268744.

Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86 (11), 2278–2324.

Lee, S., Zhang, T., Prakash, S., Niu, Y., Avestimehr, S., 2024. Embracing federated learning: Enabling weak client participation via partial model training. IEEE Trans. Mob. Comput. 23 (12), 11133–11143.

Li, T., Sahu, A.K., Talwalkar, A., Smith, V., 2020a. Federated learning: Challenges, methods, and future directions. IEEE Signal Process. Mag. 37 (3), 50–60.

Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V., 2020b. Federated optimization in heterogeneous networks. In: Dhillon, I., Papailiopoulos, D., Sze, V. (Eds.), Proceedings of Machine Learning and Systems. Vol. 2, pp. 429–450, [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2020/file/1f5fe83998a09396ebe6477d9475ba0c-Paper.pdf.

Li, E., Zhou, Z., Chen, X., 2018. Edge intelligence: On-demand deep learning model co-inference with device-edge synergy. In: Proceedings of the 2018 Workshop on Mobile Edge Communications. MECOMM '18, Association for Computing Machinery, New York, NY, USA, pp. 31–36, [Online]. Available: https://doi.org/10.1145/3229556.3229562.

Liang, C., Liang, W., Chen, Y., Li, K., Zomaya, A.Y., 2025. FedTCTF: Tensor completion-based federated learning for device heterogeneity. IEEE Trans. Sustain. Comput. 10 (6), 1227–1239.

Liang, W., Yang, Y., Yang, C., Hu, Y., Xie, S., Li, K.-C., Cao, J., 2023. PDPChain: A consortium blockchain-based privacy protection scheme for personal data. IEEE Trans. Reliab. 72 (2), 586–598.

Lim, W.Y.B., Luong, N.C., Hoang, D.T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., Miao, C., 2020. Federated learning in mobile edge networks: A comprehensive survey. IEEE Commun. Surv. Tutorials 22 (3), 2031–2063.

Liu, J., Jia, J., Che, T., Huo, C., Ren, J., Zhou, Y., Dai, H., Dou, D., 2024. FedASMU: Efficient asynchronous federated learning with dynamic staleness-aware model update. Proceedings of the AAAI Conference on Artificial Intelligence 38 (12), [Online]. Available: https://par.nsf.gov/biblio/10538070.

Liu, Y., Li, X., Zhang, C., Li, B., 2022. No one left behind: Inclusive federated learning over heterogeneous devices. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. KDD, pp. 1166–1176.

Liu, L., Zhang, J., Song, S., Letaief, K.B., 2020. Client-edge-cloud hierarchical federated learning. In: ICC 2020 - 2020 IEEE International Conference on Communications. ICC, pp. 1–6.

Luo, S., Chen, X., Wu, Q., Zhou, Z., Yu, S., 2020. HFEL: Joint edge association and resource allocation for cost-efficient hierarchical federated edge learning. IEEE Trans. Wirel. Commun. 19 (10), 6535–6548.

Luo, B., Xiao, W., Wang, S., Huang, J., Tassiulas, L., 2022. Tackling system and statistical heterogeneity for federated learning with adaptive client sampling. In: IEEE INFOCOM 2022 - IEEE Conference on Computer Communications. pp. 1739–1748.

Ma, Q., Liu, J., Jia, Q., Zhou, X., Hu, Y., Xie, R., 2024. Dynamic staleness control for asynchronous federated learning in decentralized topology. In: Wireless Artificial Intelligent Computing Systems and Applications: 18th International Conference, WASA 2024, Qindao, China, June 21–23, 2024, Proceedings, Part II. Springer-Verlag, Berlin, Heidelberg, pp. 99–117, [Online]. Available: https://doi.org/10.1007/978-3-031-71467-2_9.

McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.y., 2017. Communication-efficient learning of deep networks from decentralized data. In: Singh, A., Zhu, J. (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. In: Proceedings of Machine Learning Research, vol. 54, PMLR, pp. 1273–1282, [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html.

Nishio, T., Yonetani, R., 2019. Client selection for federated learning with heterogeneous resources in mobile edge. In: ICC 2019 - 2019 IEEE International Conference on Communications. ICC, pp. 1–7.

Nunes, A.L., Boeres, C., Drummond, L.M.A., Pilla, L.L., 2024. Optimal time and energy-aware client selection algorithms for federated learning on heterogeneous resources. In: 2024 IEEE 36th International Symposium on Computer Architecture and High Performance Computing. SBAC-PAD, pp. 148–158.

Qiao, D., Guo, S., Zhao, J., Le, J., Zhou, P., Li, M., Chen, X., 2024. ASMAFL: Adaptive staleness-aware momentum asynchronous federated learning in edge computing. IEEE Trans. Mob. Comput. 1–16.

Sprague, M.R., Jalalirad, A., Scavuzzo, M., Capota, C., Neun, M., Do, L., Kopp, M., 2019. Asynchronous federated learning for geospatial applications. In: Monreale, A., Alzate, C., Kamp, M., Krishnamurthy, Y., Paurat, D., Sayed-Mouchaweh, M., Bifet, A., Gama, J., Ribeiro, R.P. (Eds.), ECML PKDD 2018 Workshops. Springer International Publishing, Cham, pp. 21–28.

Sun, S., Zhang, Z., Pan, Q., Liu, M., Wang, Y., He, T., Chen, Y., Wu, Z., 2024. Staleness-controlled asynchronous federated learning: Accuracy and efficiency tradeoff. IEEE Trans. Mob. Comput. 23 (12), 12621–12634.

Verma, D., Julier, S., Cirincione, G., 2018. Federated AI for building AI solutions across multiple agencies. [Online]. Available: https://arxiv.org/abs/1809.10036.

Wang, J., Liu, Q., Liang, H., Joshi, G., Poor, H.V., 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20, Curran Associates Inc., Red Hook, NY, USA.

Wang, S., Ruan, Y., Tu, Y., Wagle, S., Brinton, C.G., Joe-Wong, C., 2021. Network-aware optimization of distributed learning for fog computing. IEEE/ACM Trans. Netw. 29 (5), 2019–2032.

Wu, W., He, L., Lin, W., Mao, R., 2021a. Accelerating federated learning over reliability-agnostic clients in mobile edge computing systems. IEEE Trans. Parallel Distrib. Syst. 32 (7), 1539–1551.

Wu, W., He, L., Lin, W., Mao, R., Maple, C., Jarvis, S., 2021b. SAFA: A semi-asynchronous protocol for fast federated learning with low overhead. IEEE Trans. Comput. 70 (5), 655–668.

Xie, C., Koyejo, S., Gupta, I., 2020. Asynchronous federated optimization. [Online]. Available: https://arxiv.org/abs/1903.03934.

Yan, X., Miao, Y., Li, X., Choo, K.-K.R., Meng, X., Deng, R.H., 2023. Privacy-preserving asynchronous federated learning framework in distributed IoT. IEEE Internet Things J. 10 (15), 13281–13291.

Yang, Q., Liu, Y., Chen, T., Tong, Y., 2019. Federated machine learning: Concept and applications. ACM Trans. Intell. Syst. Technol. 10 (2), [Online]. Available: https://doi.org/10.1145/3298981.

Yao, X., Sun, L., 2020. Continual local training for better initialization of federated models. In: 2020 IEEE International Conference on Image Processing. ICIP, pp. 1736–1740.

Zeng, S., Li, Z., Yu, H., He, Y., Xu, Z., Niyato, D., Yu, H., 2022. Heterogeneous federated learning via grouped sequential-to-parallel training. In: Database Systems for Advanced Applications: 27th International Conference, DASFAA 2022, Virtual Event, April 11–14, 2022, Proceedings, Part II. Springer-Verlag, Berlin, Heidelberg, pp. 455–471, [Online]. Available: https://doi.org/10.1007/978-3-031-00126-0_34.

Zhang, S., Hu, B., Liang, W., Li, K.-C., Gupta, B.B., 2023. A caching-based dual K-anonymous location privacy-preserving scheme for edge computing. IEEE Internet Things J. 10 (11), 9768–9781.

Zhang, S., Ren, F., Liang, W., Li, K., Ling, N., 2025. GPVO-FL: Grouped privacy-preserving and verification-outsourced federated learning in cloud-edge collaborative environment. IEEE Trans. Netw. Serv. Manag. 22 (5), 4175–4191.

Zhang, S., Yan, Z., Liang, W., Li, K.-C., Di Martino, B., 2024. BCAE: A blockchain-based cross domain authentication scheme for edge computing. IEEE Internet Things J. 11 (13), 24035–24048.

Zhao, Z., Feng, C., Hong, W., Jiang, J., Jia, C., Quek, T.Q.S., Peng, M., 2022. Federated learning with Non-IID data in wireless networks. IEEE Trans. Wirel. Commun. 21 (3), 1927–1942.