Expert Systems With Applications 281 (2025) 127636

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Accelerating Federated Learning with genetic algorithm enhancements

Huanqing Zheng [a], Jielei Chu [a,*], Zhaoyu Li [a,b,c], Jinghao Ji [a], Tianrui Li [a]

[a] School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, 611756, China
[b] Technological Innovation and Digitalization Department, China Railway Engineering Group Limited, Beijing, 100039, China
[c] CREC R&D Center for Carbon Peaking and Carbon Neutrality, China Railway Eryuan Engineering Group Co., Ltd., Chengdu, 610031, China

ARTICLE INFO

Keywords:
Federated learning
Genetic mechanism
Aggregation method

ABSTRACT

Federated Learning (FL) enables collaborative model training across multiple devices while preserving data privacy. However, developing robust and efficient FL faces significant challenges, such as data heterogeneity, computational resource constraints, communication bottlenecks, and the presence of malicious participants. To address these issues, we introduce GenFed, an innovative framework that enhances federated learning through genetic algorithm mechanisms. GenFed optimizes model aggregation strategies and balances resource utilization, thereby improving performance and resilience. This framework is designed for seamless integration with existing FL systems, facilitating rapid adaptation. GenFed accelerates model convergence and enhances robustness, particularly in environments with a large number of clients. Experimental results demonstrate that GenFed significantly outperforms traditional FL methods in terms of convergence speed, accuracy, and resilience against adversarial attacks across diverse datasets. Notably, as the number of clients increases, conventional federated methods often suffer substantial performance degradation. In contrast, GenFed maintains stable, high-level performance, making it especially practical for real-world scenarios involving extensive client participation. Our findings indicate that GenFed is a versatile and efficient solution that offers significant improvements in scalability and robustness, contributing to the deployment of reliable federated learning in real-world applications.

1. Introduction

FL (Li, Sahu, Talwalkar, & Smith, 2020; Yang, Liu, Chen, & Tong, 2019) enables diverse participants to engage in the training process without the exchanging raw data with its decentralized architecture. This methodology substantially enhances model accuracy and robustness by integrating diverse data sources, while it simultaneously ensures strict adherence to data privacy and security measures. Consequently, FL has been rapidly embraced in sectors with high data sensitivity and limited data availability, such as healthcare (Ali Khowaja, Dev, Muhammad Anwar, & George Linguraru, 2025; Damiani, Rodina, & Decherchi, 2024), finance (Imteaj & Amini, 2022), and telecommunications (Al-Huthaifi et al., 2024; Moreno-Álvarez, Paoletti, Sanchez-Fernandez, Rico-Gallego, Han, & Haut, 2024; Song et al., 2021). The capability to leverage data within a framework of rigorous privacy standards has established FL as a cornerstone technology, advancing machine learning in these critical fields.

However, FL faces many challenges such as slow training speed, high resource consumption, and susceptibility to the influence of extreme clients (Li et al., 2020). Firstly, the model aggregated and distributed by the server is susceptible to deviations from the correct gradient direction due to issues such as client drift, which frequently arise from the diversity in data distribution across clients (Gao et al., 2022; Karimireddy, Kale et al., 2020). So, comparing with centralized learning, FL need more training rounds to achieve the same level performance. Secondly, due to the inherently slow training process and the substantial communication required in each training round, FL encounters another significant challenge: resource consumption (Hamer, Mohri, & Suresh, 2020). This encompasses not only communication resources but also computational resources and time costs on both the server and the clients (Luo, Li, Wang, Huang, & Tassiulas, 2021). Thirdly, during the FL process, clients are not consistently reliable and may not always contribute positively (So, Güler, & Avestimehr, 2020). Clients with malicious intent or highly skewed data distributions can adversely affect the convergence of the global model, thereby complicating the achievement of an optimal state.

Existing research predominantly focuses on identifying suitable aggregation algorithms to address the aforementioned issues (Qi et al., 2023). Regarding the issue of slow model training, dropping data from slow clients is a common method for accelerating training in

* Corresponding author.
E-mail addresses: gswj2001@my.swjtu.edu.cn (H. Zheng), jieleichu@swjtu.edu.cn (J. Chu), lizhaoyu@swjtu.edu.cn (Z. Li), jjh@swjtu.edu.cn (J. Ji), trli@swjtu.edu.cn (T. Li).

https://doi.org/10.1016/j.eswa.2025.127636
Received 18 November 2024; Received in revised form 27 February 2025; Accepted 7 April 2025
Available online 17 April 2025

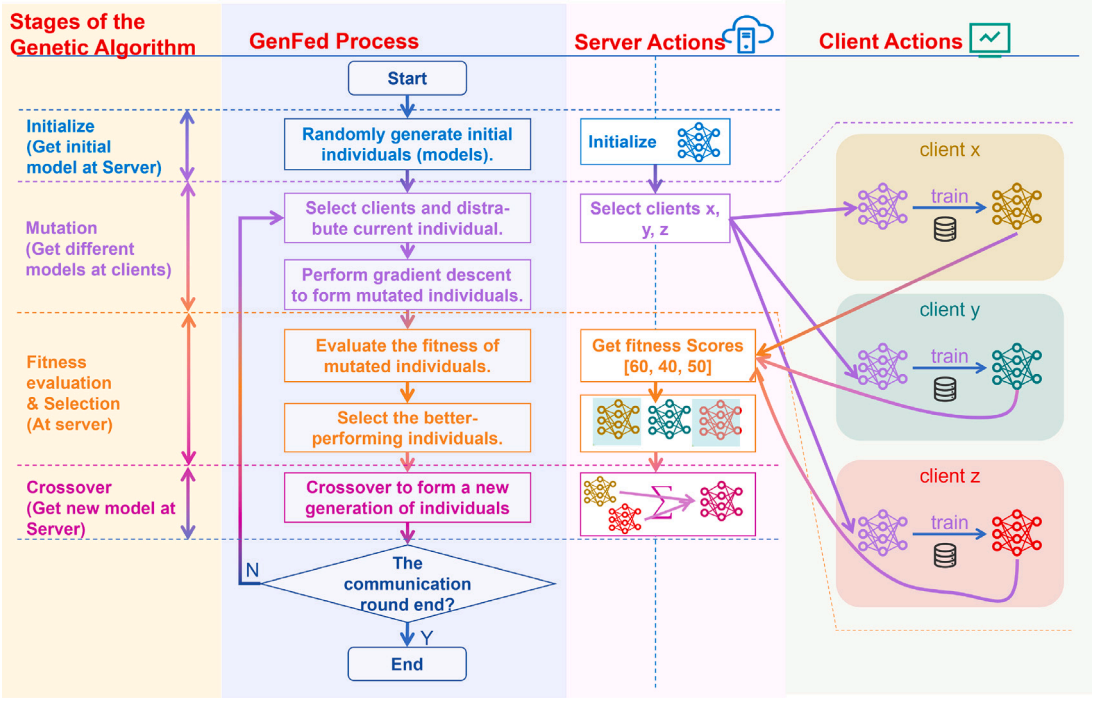

Fig. 1. Genetic mechanisms perspective of GenFed methodology. Initialize, Mutation and Crossover are processes that exist in traditional federated learning algorithms, while Fitness evaluation & Selection is a process unique to our GenFed algorithm. For illustration purposes, the number of participating clients (3), the fitness scores of the models ([60, 40, 50]), and the number of models selected for aggregation per round (2) are examples to facilitate understanding. In practical implementations, the number of participating clients is typically much larger, and in our design, the number of models selected for aggregation varies dynamically each round.

synchronous FL (Lee, Ko, & Pack, 2021; Liu et al., 2021; Wu & Wang, 2022). However, it is unfair that directly and aggressively discards data from low-performance clients. Additionally, these clients might possess a significant amount of unique data. Their consistent exclusions from the training process may lead to the model converging to a suboptimal state. Regarding the issue of high resource consumption, reducing the amount of data transmitted in each round of communication can effectively decrease the per-communication volume in a FL architecture (Liang, Zhao, Liang, Wu, & Guo, 2024; Rahimi, Bhatti, Park, Kousar, & Moon, 2024; Zhang et al., 2025). However, such approaches may lead to data insufficiency, resulting in fluctuations in the training process. Consequently, the model requires more training rounds to achieve the same level of performance compared to traditional schemes where the communication content consists of model parameters or model gradients. In other words, both the client and server require longer training times and more computational resources. Regarding the issue of model robustness, model pruning and regularization are the most commonly used methods (Karimireddy, He, & Jaggi, 2024; Murata, Niwa, Fukami, & Tyou, 2024; Pillutla, Kakade, & Harchaoui, 2022). The core of these approach is how to reduce the disparity of the global model between rounds. However, such approaches may risk leading model training into local optima. That is why they always typically face more severe issues of slow training. Currently, several algorithms attempt to enhance model robustness by using greedy strategies (Cong et al., 2024). In conclusion, existing methods struggle to achieve an optimal balance among training speed, resource consumption, model robustness, and model performance.

Unlike previous work, we analyze the FL architecture from a novel perspective and accordingly design a new FL aggregation algorithm to accelerate the acquisition of high-performing models. In the context of FL, we assume that the server operates in a fair environment, which can evaluate models from clients fairly and effectively. Conversely, the client devices are distributed in diverse environments, each with its unique characteristics. During the FL process, if a set of model parameters receives a better evaluation on the server side, this set of model parameters is likely to exhibit higher accuracy across various client environments. In other words, if a set of model parameters demonstrates a higher survival rate in the server "environment", it is expected to perform better in client environments as well. This premise forms the basis of our discussions. From the perspective of genetic mechanisms, we find that:

- The aggregation process on the server is similar to the genetic process of **crossover**, which produces the subsequent generation of a population.
- The process of model training and optimization on the client side can be viewed as a form of **mutation**, introducing variability within the model parameters.
- The distinction between FL and genetic algorithms lies in the absence of a fitness scoring mechanism for model **selection** in FL.

Traditional genetic algorithms select individuals with higher fitness scores within a generation to engage in crossover, providing the foundation for variation in subsequent generations. How to introduce such a mechanism into FL? A novel methodology should be devised to construct fitness functions, which align with the principles of fairness and equity inherent in the server environment. The server can select models from the clients which have demonstrated higher fitness scores, then they are aggregated to form a new generation of models through a crossover mechanism.

In this work, we apply above genetic mechanisms to optimize FL methods. By designing a reasonable model selection mechanism (i.e. suitable fitness function) on the server side, we get the federated learning training process shown in Fig. 1, which can be strictly compared with the genetic mechanism. Through such training process, our algorithm achieves the purpose of accelerating training. The main contributions of this paper are as follows:

- Architectural Innovation: We propose a novel approach for FL by rethinking and reconstructing the architecture from a genetic mechanism perspective.

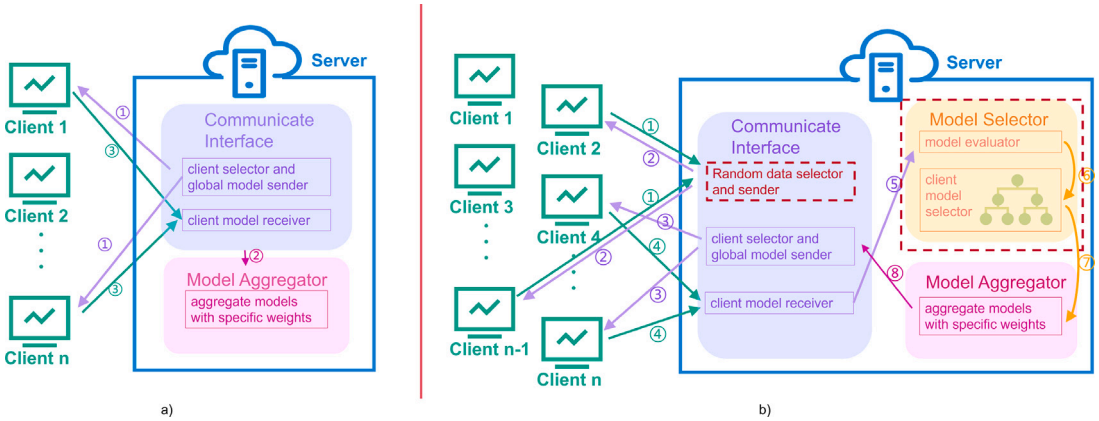

Fig. 2. Comparison of frameworks: (a) illustrates the traditional federated learning framework, while (b) depicts the proposed GenFed framework. The proposed methodology introduces supplementary components (the parts outlined in red in the diagram) to the existing framework, which are orthogonal to the intrinsic optimizations of the framework itself.

- Aggregation Scheme: We innovatively propose a plug-and-play aggregation scheme that the server selectively filters high-quality participant data to update the global model.
- Empirical Validation: Experimental findings has demonstrated that our GenFed approach, when incorporated as a lightweight modification to existing methods, can not only accelerate convergence and improves performance on comparable datasets but also enhance model robustness. This makes the model more stable and reliable, even in the presence of unreliable or heterogeneous clients. Furthermore, the GenFed method has shown its capability to mitigate the performance degradation typically associated with a high number of clients, which is of considerable significance in real-world applications.

2. Related work

2.1. Traditional FL framework

FL is a methodology which enhances local model performance by facilitating joint training through communication between clients and the server, all while safeguarding the privacy of sensitive data residing on the clients' side (McMahan, Moore, Ramage, Hampson, & y Arcas, 2017). The typical process of FL is illustrated in Fig. 2(a). The three steps of the framework are as follows: ① The server randomly selects clients to participate in the current round and distributes the current global model w_g^t, which can be regarded as the selection of aggregation objects. ② Each client i in the set of selected clients C_t performs local epochs of gradient descent on the received global model and send its trained models w_i^t back to the server, which can be regarded as the generation of aggregation content. ③ The server aggregates the models received from the clients and updates the global model to w_i^{t+1}. In the step of selecting aggregation targets, the focus is primarily on alleviating communication congestion, without considering the quality of the aggregation content itself. In contrast, our design incorporates an evaluation of the quality of the aggregation content on the server side before aggregation. Specifically, our approach involves selecting the higher-quality content for aggregation, rather than aggregating all available content from participants. This method can improve the quality of the aggregation results to some extent.

2.2. Aggregation algorithm in FL

In FL, aggregation algorithms constitute a fundamental component, tasked with synthesizing local model updates from disparate clients into a cohesive global model. The model aggregator illustrated in Fig. 2(a) is responsible for executing this integration process. Asynchronous FL (AFL) and Synchronous FL (SFL) are two primary modes for updating the model. AFL allows the clients to update asynchronously (Xu, Qu, Xiang, & Gao, 2023), which accelerates the training process by preventing slower clients from impeding overall progress (Jiang, Lu, Mao, & Lin, 2023; Liu et al., 2024; Nguyen et al., 2022), thus enhancing the system's flexibility and scalability. However, it may lead to the problem of outdated model updates. Some clients might potentially perform updates based on an outdated global model. To address this issue, researchers have proposed various strategies, such as discarding outdated models (Sun, Shao, Mao, & Zhang, 2022), dynamically adjusting the local training workload (Zhang, Gao, Lee, Zhang, & Avestimehr, 2023) and performing time-weighted aggregation for model updating (Cheng, Huang, Wu, & Yuan, 2024; Huba et al., 2022; Zang et al., 2024). Nevertheless, the impact of model consistency issues on the global model remains substantial. So our work focuses on synchronous aggregation methods, which always provide superior model performance.

In SFL, the server must receive a sufficient number of client models in order to perform this round of aggregation (Qi et al., 2023). By incorporating efficient updating mechanisms, which significantly accelerate the training process while maintaining model performance, the method has been further extended. FedPA (Liu et al., 2021) is designed such that the server waits for only a predetermined number of device models, as specified by server-side agents in each round, before aggregating them to obtain the new global model. Under similar considerations, (Lee et al., 2021) formulated an adaptive selection mechanism for the determination of the temporal threshold and the cohort of mobile devices engaged in the training iteration. But both of them ignored the value of low-performing client data, which can negatively impact model performance. Other researchers argue that a more global model can be obtained by selecting clients that participate or selecting models aggregate in each round for each round based on a certain strategy. Eiffel (Sultana, Haque, Chen, Xu, & Yuan, 2022) assigned an index based on client performance to indicate the priority of each participating client selected by the scheduling algorithm. FedAAW (Yin et al., 2024) optimizes aggregation weights using client gradients and synchronizes weight updates across local models, thereby hastening model convergence. FedAU (Wang & Ji, 2024) adaptively used aggregate weights that were inversely proportional to the clients' average participation rates, thereby enhancing the model's accuracy. The above scheme of filtering aggregate clients based on evaluation is very unfair to some clients. And some outside hackers may poison some clients more targeted, resulting in a decline in the overall training effect. However, in the selection scheme based on evaluation, due to the designed evaluation criteria, the number of storage clients in each round of screening is fixed and close to each other, and some effective client data may be lost, resulting in slow training. Our work is also

Table 1
Notations and descriptions.

Notation	Description
D_g^v	Global validation dataset
D_i	Dataset at client i
D_i^e	Supplementary data at client i
w_g^t	Global model at time t
w_i^t	Model at client i at time t
t_{wait}	Server-side waiting time for client responses during the preprocessing stage
p_i	Aggregation weight of the model from client i
ρ_t	Number of models participating in aggregation at time t
ρ_{max}	Maximum value of the aggregation quantity
$\mathcal{K}$	Number of clients participating in training per round
S_t	Set of clients participating in training at time t
C_t	Set of clients participating in aggregation at time t
c	Maximum number of variation iterations for the aggregation quantity
r	The number of rounds in which the participating clients perform local training
$\mathcal{O}_g$	Computational overhead at the server side per communication round
$\mathcal{O}_i$	Computational overhead at client i per communication round
α	Dirichlet distribution parameter for data heterogeneity
$\mathcal{M}$	Model Parameter Transmission Volume

about selective aggregation of aggregation models, but the screening criteria is different from the previous works. And experimental verification shows that our work can effectively accelerate training process even in the presence of poisoners, which is our advantage over similar methods.

2.3. Genetic mechanisms

In genetic algorithms, potential solutions of a problem are encoded as chromosomes (Shah & Bichkar, 2021), which might take the form of binary strings, real-valued vectors, or other representations. The algorithm begin by randomly generating an initial population of chromosomes. Each chromosome is assigned a fitness value based on the quality of its solution, which serves as an indicator of its effectiveness in solving the problem. Chromosomes are selected for reproduction based on their fitness values (Albadr, Tiun, Ayob, & Al-Dhief, 2020), with those having higher fitness being more likely to be chosen (Michalewicz & Schoenauer, 1996). Selected chromosomes undergo crossover operations (Umbarkar & Sheth, 2015) with a certain probability to produce new offspring chromosomes. To introduce new genetic information and maintain population diversity, chromosomes are subjected to random mutations with a relatively low probability (Deng, Xu, Song, & Zhao, 2021). The new chromosomes generated through crossover and mutation form the next generation of the population. This process is repeated until stopping criteria are met, such as reaching a maximum number of iterations or achieving a solution quality that meets a predefined threshold (Alam, Qamar, Dixit, & Benaida, 2020).

Recent studies have explored the use of genetic algorithms in various aspects of federated learning. Kang and Ahn (Kang & Ahn, 2023) proposed optimizing client selection through genetic algorithm, with participants in each training round represented as chromosomes. The fitness of each chromosome is evaluated based on model divergence and communication costs. However, this approach risks unfair client selection, as some clients may be overrepresented while others are underrepresented due to communication costs. Moreover, it does not address adversarial behaviors, such as selecting clients with significant model divergence, which could lead to performance degradation. In contrast, our method avoids relying on a fixed set of clients and is more robust to such attacks by diversifying the client selection process.

Guendouzi, Ouchani, and Malki (2022) introduced FedGA-ICPS, which uses genetic algorithm to optimize model aggregation weights in industrial cyber–physical systems. Although our method also integrates genetic algorithm into federated learning, we focus on selecting models for aggregation rather than fine-tuning aggregation weights. This selective model aggregation accelerates the training process and can be combined with FedGA-ICPS to further enhance training speed.

In vehicular networks, Khatua, Mukherjee, and De (2024) applied genetic algorithm to optimize route selection for federated learning. Unlike our framework, which focuses on client-side decision-making (e.g., classification tasks), their approach optimizes server-side decisions, specifically selecting the optimal routes for vehicles based on real-time data. Although both methods employ genetic algorithm, they address different aspects of federated learning and are complementary.

Finally, De Falco et al. (2023) proposed FLEA, an evolutionary algorithm inspired by federated learning for glucose prediction in healthcare. FLEA encodes models as chromosomes but requires all clients to participate in every round and share models with all other clients. This results in significant communication overhead, particularly in large-scale federated systems. In contrast, our approach reduces communication bottlenecks by not requiring full client participation in each round.

3. Methodology

Table 1 summarizes the notations and descriptions used throughout this paper.

The overall framework and communication process of GenFed are illustrated in Fig. 2(b). The entire procedure can be divided into two phases: initialization and training. The initialization phase has two steps: ① Clients with insufficient data send data supplementation request to the server. ② The server randomly selects a small amount of data from the global validation set to supplement the requesting clients. The training phase has six steps: ③ The server randomly selects active clients for the current round and dispatches the current global model w_g^t. ④ Clients receive the global model, perform gradient descent for local epochs rounds, and return the trained model to the server. ⑤ The server evaluates the global model w_i^t returned by client i based on its performance on the global validation set and assigns a score. ⑥ Using these scores, the server employs a binary heap data structure to quickly select the top ρ_t performing models. ⑦ The selected models are aggregated to form a new global model w_g^{t+1}. ⑧ Steps 3 to 8 are repeated until the model achieves the specified accuracy or the total number of communication rounds is exhausted. Since a considerable amount of existing FL aggregation algorithms are developed for steps 3, 4, 7, or 8, our work and these studies are orthogonal. This also means that our work can easily be extended to use existing research results to achieve better results.

3.1. Global validation set

In our work, the main change of the Fl framework is based on the genetic mechanisms. The mapping of genetic mechanisms and FL architectures puts the following requirements for the implementation of FL architecture:

- The server should be able to fairly represent the client's environment. Conversely, the emergence of an unbalanced global environmental context may precipitate server-side biases, thereby inducing a directional bias in the evolution of the global model. This phenomenon can significantly impede the model's capacity for robust generalization across a spectrum of data scenarios.
- The server needs to evaluate the models sent back by the client based on this representation, and accordingly select the set of models participating in the aggregation. In the absence of a robust model evaluation process, enhancing the training process and mitigating or precluding the influence of aberrant models within the aggregation phase becomes infeasible.

Considering the aforementioned factors, we have established a publicly accessible dataset, which is characterized by a balanced distribution and devoid of privacy constraints, designed to function as the global validation set. This dataset serves as a pivotal resource during the training phase, providing clients with a standardized benchmark for model evaluation. The server employs this dataset to rigorously evaluate the predictive accuracy of the models returned by the client in each training round. Subsequently, models demonstrating superior performance are meticulously selected and integrated into the aggregation process to form a new global model.

While the foundational principles of FL emphasize privacy preservation, the creation of a global validation set might raise concerns about potential conflicts with these principles. The core of this dilemma lies in whether utilizing a global validation set infringes upon the privacy rights of client data. However, since our approach relies on publicly available datasets rather than data directly sourced from clients for global validation, this concern is effectively mitigated. For example, in machine learning, disease detection through chest X-rays often uses the CheXpert dataset (Irvin et al., 2019), recognized for its privacy-neutral characteristics. Nevertheless, the confidentiality of patient radiographic data remains critical, necessitating stringent data protection measures by medical institutions as part of their compliance with federal regulations. But by employing our method, the CheXpert dataset can be utilized as a validation resource on the server side, ensuring that client-side data remains unaffected. This approach, therefore, upholds the integrity of privacy protections at the client level.

3.2. Data supplement request

In the process of FL, certain clients, due to their limited data sets, are prone to overfitting, a phenomenon that can adversely affect the performance of the overarching global model (Li, He, & Song, 2021). To mitigate this issue, an initialization phase has been introduced prior to the commencement of the training phase.

At the beginning of the initialization phase, each client evaluates the amount of its own data. If the amount of its own data is too small, it will send data supplement request to the server. And at the same time, the server wait for a fixed time t_{wait}. After that, it randomly select a portion of data in the global verification data set for each client who submitted request independently, and send it to the corresponding client. And after that, the training phase will start after t_{wait}, which is the reserved time used to wait for the clients to receive supplementary data.

From the clients' perspective, non-malicious entities are expected to engage in this phase with integrity and proactivity, contributing constructively to the process. Clients possessing a limited dataset are likely to witness a considerable boost in their model's efficacy through the server's data provision. In contrast, those with a substantial dataset might find that the server's data contributes minimally to their model's enhancement and could even lead to an increased demand on their computational resources. Therefore, for these clients, engaging in the process with deceit is not only unnecessary but could also be counterproductive. On the other hand, clients with malicious intent may view the server's data distribution as an opportunity to initiate an attack, using the obtained data to further their harmful agenda. However, since the server randomly selects a portion of the data to send to the clients, for any client i, j that make this request, the supplementary data they receive, D_i^e, D_j^e, and the global data D_g^v satisfy the following relationship:

$$D_i^e \neq D_j^e \ \wedge \ |D_i^e| \ll |D_g^v|, \tag{1}$$

which means that client with malicious intent cannot acquire not only complete global validation data but also the same data that any other clients obtained. This causes a certain level of difficulty for their attack behavior at this stage. Meanwhile, since the initialization phase only occurs once and takes up a very small proportion of the overall time in our framework, defending against attacks at this stage is not the primary focus of this study.

From the server's perspective, the use of publicly available, privacy-agnostic data as the global validation data has already been discussed in the Global Validation Set section. Consequently, the one-time data supplementation during this phase is not considered to compromise the overarching privacy safeguards of the system.

Algorithm 1 Selecting Optimal Subset and Aggregating Global Model

Input: Global validation set D_g^v, set of participating clients C, subset size ρ_t

Parameter: an empty binary heap H to store client-score pairs (i, Score_i), $|H|$ denote the number of elements contained within H

Output: Updated global model ω_g^{t+1}

function HeapifyUp($H, index$)

```
while index > 0 and H[⌊index/2⌋] < H[index] do
    Swap H[⌊index/2⌋] with H[index].
    index ← ⌊index/2⌋.
```

main procedure

```
for each client i ∈ C do
    Evaluate the model of i on D_g^v to obtain evaluation score Score_i.
    if |H| < ρ_t then
        Insert the client-score pair (i, Score_i) into the next available position of H.
        HeapifyUp(H, |H| − 1).
    else if Score_i > Score of top element in H then
        Swap the top element and the last element of H.
        Remove the last element from H.
        Insert the client-score pair (i, Score_i) into H.
        HeapifyUp(H, |H| − 1).
Aggregate the models of the top ρ_t clients in the heap to form the updated global model ω_g^{t+1}
```

3.3. Aggregation object selection per round

In traditional FL framework, all models received by server are aggregated to get the new global model. The mathematical expression is given by

$$w_g^{t+1} = \sum_{i \in S_t} p_i w_i^t, \tag{2}$$

where S_t represents the set of clients chosen to participate in this communication round; p_i represents the aggregate weight of model from client i.

But in our design, a core component is the model selection component. The function of the component is that in each communication round, the server side evaluates the model sent by the client and selects the best performing ρ_t models by using global validation set and add them to a new set C_t. It is straightforward to deduce that C_t and S_t are related as,

$$C_t \in S_t. \tag{3}$$

Then the new global model will be aggregated from the models in C_t using the formula as,

$$w_g^{t+1} = \sum_{i \in C_t} p_i w_i^t. \tag{4}$$

In other words, we use the optimal subset of the set of participated clients with size ρ_t on the global validation set to aggregate, than obtain a new global model.

Specifically, an optimal subset of participating clients, with cardinality ρ_t, is selected based on performance on the global validation set to contribute to the aggregation, thereby yielding an updated global

model. The evaluation and ranking of clients is significantly expedited by utilizing a balanced binary heap data structure, as detailed in Algorithm 1. Given $\mathcal{K}$ participating clients, this selection module operates with a time complexity of $\mathcal{O}(\mathcal{K}\log\mathcal{K})$. In comparison to the computational burden of local training and communication, this additional cost is minimal and remains comfortably within acceptable limits. Furthermore, considering that numerous FL optimization algorithms enhance model performance by adjusting the weights of aggregated models, this design can be seamlessly integrated to achieve improved outcomes.

A fundamental aspect of the aforementioned design is the number of models utilized for aggregation in each round, which can be denoted as ρ_t. In our work, various strategies for adjusting the aggregation quantity are explored to investigate their effects on system behavior. Initially, the baseline approach maintains the aggregation quantity at a constant value, denoted as,

$$\rho_t = \rho_{max}, \tag{5}$$

where ρ_{max} represents the maximum value of the aggregation quantity within the experimental setup. Beyond this, a power function is employed to introduce variability, with the aggregation quantity modeled as,

$$\rho_t = \rho_{max} * (1 - b^t) + 1 \text{ while } \ b \in (0, 1), \tag{6}$$

where b represents the base of the power function which need to be set before the experiment. A linear variation is also considered, where the aggregation quantity evolves according to the expression,

$$\rho_t = \min(\rho_{max} * \frac{t}{c} + 1, \rho_{max}), \tag{7}$$

where c represents the maximum number of variation iterations for the aggregation quantity. Additionally, we introduced sinusoidal variations in the aggregation quantity. The first sinusoidal model follows a sine function with a period of $\frac{\pi}{2}$, which is described by,

$$\rho_t = \min(\rho_{max} * \sin\frac{t}{2c}\pi + 1, \rho_{max}). \tag{8}$$

Another model extended this approach to a sine function with a full period of π, where the aggregation quantity is expressed as,

$$\rho_t = \begin{cases} \rho_{max} * \sin\frac{t}{c}\pi + 1 & , t < c \\ 1 & , t \geq c. \end{cases} \tag{9}$$

In the experimental section, we will discuss the configurations of the aforementioned different methods for varying the aggregation quantity.

3.4. The full GenFed algorithm with computational overhead analysis

Algorithm 2 presents a detailed account of the GenFed framework, with annotations delineating the correspondence between the genetic mechanisms and the federated process. The algorithm is executed in two main phases: Initialization and Training, with distinct procedures for both the server and client sides. During the initialization phase, the server and clients complete a single round of communication, enabling the server to offer targeted support to clients with extreme data volumes. This strategy helps to address the challenges posed by data heterogeneity among clients. In the subsequent training phase, the server employs a genetic algorithm-inspired approach, selectively aggregating information from certain clients to accelerate the training process. Furthermore, this method helps to filter out potentially malicious data from compromised clients.

In the process described above, we consider the computational cost incurred at the client side. Our framework incurs the same per-round computational overhead at the client side as other federated learning frameworks. Specifically, in our approach, each client performs r rounds of local training. Therefore, the time complexity of the computation at client i during this parallel execution is expressed as:

$$\mathcal{O}_i = \mathcal{O}(r\mathcal{M}), \tag{10}$$

Algorithm 2 GenFed

Input: Global validation set D_g^v, local datasets D_i for each client i
Parameter: an empty binary heap H to select optimal subset, server wait time t_{wait}, number of training rounds T, the maximum value of the aggregation quantity ρ_{max}
Output: the final global model ω_g^{t+1}
Server-Side Execution:
Initialization Phase:

Initialize server model parameters w_g^0 and the aggregate number ρ_0 at the first round.
Wait t_{wait} for receive data supplement from clients.
for each request r received **do**
    Randomly select a portion of global data.
    Send the data to client r_k according to request r.
Wait t_{wait} for the clients to receive supplementary data.

Training Phase:

for each round $t = 1, 2, \ldots, T$ **do**
    Select a subset of clients S_t
    for each client $k \in S_t$ **in parallel do**
        Send the current model parameters w_g^t to client k.
        Client k performs local training and returns updated parameters w_k^t.
▷ *MUTATION*
    Evaluate and choose the best ρ_t models from all received models.
▷ *SELECTION*
    Update the server model parameters w_g^{t+1} using the best ρ_t models.
▷ *CROSSOVER*
    Replace ρ_t with ρ_{t+1} to prepare for the next communication round.

Client-Side Execution:
Initialization Phase:

if the dataset is of a sufficiently limited scope **then**
    Send data supplement request to server.
    Receive data and add it to the train set.

Training Phase:

Receive the model parameters w_g^t from the server.
Perform local training on the client dataset and update local model parameters w_k^t.
Send the updated parameters w_k^t back to the server.

where r denotes the number of local training rounds conducted by the client during each communication round, and $\mathcal{M}$ represents the size of the model parameters. Since the model size is uniform across clients, the execution speed at each client is primarily determined by the size of the local dataset and the computational capacity of the client's machine.

Next, we examine the computational cost at the server side. In our federated learning framework, the server first evaluates the received models on the global validation set in each communication round. The computational cost of this evaluation phase is given by $\mathcal{O}(K\mathcal{M})$, where K represents the number of participating clients. Following this, the server ranks the models based on their performance and selects the top-performing models for averaging. The computational cost of this ranking process, as previously analyzed, is $\mathcal{O}(K\log K)$. Finally, the server aggregates the models from the top ρ_t clients, which incurs a complexity of $\mathcal{O}(\rho_t\mathcal{M})$. Since the values of K and ρ_t are relatively small compared to the model size $\mathcal{M}$, which is typically much larger, the overall computational overhead per communication round at the server is dominated by the term $\mathcal{O}(K\mathcal{M})$. As a result, the total computational cost at the server side per communication round can be approximated

as:

$$\mathcal{O}_g < \mathcal{O}(2\mathcal{K}\mathcal{M}). \tag{11}$$

This computational cost is similar to that of traditional federated learning frameworks at the server side.

However, when considering the entire process, our framework significantly accelerates the training procedure, reducing the number of communication rounds required to achieve high performance. As a result, the overall computational cost, both at the client side and the server side, is substantially lower with our approach.

4. Experiments

4.1. Experiment settings

4.1.1. Dataset

In the conducted experiments, four datasets are utilized: MNIST (LeCun, 2010), SVHN (LeCun et al., 2011), FashionMNIST (Xiao, Rasul, & Vollgraf, 2017), and CIFAR-10 (Deng et al., 2021). Details of these datasets are as follows:

- **MNIST:** The MNIST, a benchmark in the field of image recognition, comprises 60,000 training images and 10,000 test images of handwritten digits, each represented as a 28 × 28 grayscale image.
- **SVHN**: SVHN: The SVHN dataset contains over 600,000 images of digits (0–9) extracted from house numbers in Google Street View. It includes 73,257 training images, 26,032 test images, and 531,131 unlabeled images, each with a size of 32 × 32 pixels.
- **FashionMNIST**: The FashionMNIST dataset, designed as a drop-in replacement for MNIST, includes an identical structure with images representing various articles of clothing and accessories, thereby presenting a more complex visual recognition challenge.
- **CIFAR-10**: The CIFAR-10 dataset, a well-established dataset in computer vision, consists of 60,000 color images in 10 classes, with 6000 images per class, each image being 32 × 32 pixels in size.

4.1.2. Environment and training setup

The experimental environment is configured with an RTX 3090 GPU to facilitate the computational requirements of the study. Both clients and the server employ homogeneous model architectures, as the primary aim is to assess the efficacy of the GenFed framework, rather than the complexity of the network itself. Consequently, simple Convolutional Neural Networks (CNNs) are employed across all datasets.

For the MNIST dataset, a basic CNN architecture is implemented. This network comprise two convolutional layers, each followed by a pooling layer. The output is then flattened and passed through two fully connected layers, culminating in the final output layer for classification. A structurally analogous CNN is utilized for the FashionMNIST dataset. In contrast, for the SVHN dataset and the CIFAR-10 dataset, a slightly more complex CNN is adopted, consisting of two convolutional and pooling layers, followed by a flattening layer, three fully connected layers, and an output layer.

To ensure a non-IID distribution (Non-Independent and Identically Distributed) across clients, the local data is partitioned using a Dirichlet distribution (Hsu, Qi, & Brown, 2019) with a parameter of $\alpha = 0.1$. The Dirichlet distribution is a multivariate probability distribution, parameterized by a vector α. In the context of federated learning, the α vector determines the relative proportion of each class within the dataset. A smaller α value indicates a more imbalanced distribution of data across clients, where certain classes might be more concentrated among a few clients; conversely, a larger α value results in a more uniform distribution of data.

The total number of clients is maintained at 100, unless explicitly stated otherwise. Importantly, there is no overlap between the clients' local data and the global validation data held by the server. During each communication round, 10 clients are randomly selected to participate in the training process. Each selected client perform local training for five epochs, after which the locally updated models are transmitted to the server. The server then aggregate these models and disseminate the updated global model back to the clients.

A learning rate of 0.001 is consistently applied throughout all experiments, coupled with a learning rate decay strategy to optimize training efficiency. Given the relative simplicity of the MNIST and FashionMNIST datasets, the total number of training rounds is set at 2000 for these datasets. Conversely, for the more complex CIFAR-10 dataset, the training duration is extended to 4000 rounds to accommodate its increased difficulty.

To obtain more reliable results, each experiment mentioned in this paper has been repeated five times with different random seeds, and the final result was taken as the mean of these repetitions.

4.1.3. Baselines

In our research, we integrate the proposed GenFed mechanism into the following federated learning methods to assess its effectiveness:

- **FedAvg** (McMahan et al., 2017): As the foundational federated learning approach, FedAvg involves uploading each device's updated model parameters to a central server in every round. The server aggregates these updates by computing their average, subsequently updating the global model and distributing it back to the devices.
- **FedUmf** (Li, Chen, & Yu, 2022): FedUmf enhances training efficiency by leveraging the computational resources of unselected clients. During each communication round, these unselected clients do not remain idle; instead, they perform parallel training alongside the selected clients. If an unselected client is chosen in a subsequent round, the locally stored gradient information is merged with the global model, and a new gradient is computed in a non-trivial manner. If the client remains unselected, the stored stochastic gradient descent is overwritten by a new one derived from the updated global model.
- **FedAU** (Wang & Ji, 2024): FedAU introduces an adaptive weighting mechanism that refines the FedAvg algorithm. Its primary objective is to prevent the global model from being disproportionately influenced by a single client due to its frequent participation. This potential bias, stemming from varying participation rates among clients, is mitigated by adjusting the update weight inversely proportional to the client's average participation rate, based on an online estimation of optimal weights.

By comparing the performance of these methods with and without the integration of GenFed, we validate the efficacy of our approach. All experiments are conducted under identical conditions, closely replicating the experimental settings outlined in the original papers that introduced these baseline methods.

4.2. Main results

4.2.1. Performance comparison across different datasets

To evaluate the performance and generalizability of our federated framework with the integrated genetic mechanism, we implemented classical algorithms within this framework. Specifically, we incorporated the FedAvg aggregation algorithm, the training acceleration method FedUmf, and the advanced FedAU approach. This resulted in the creation of new variants: GenFed, GenFedUmf, and GenFedAU. We evaluated the performance of these methods using three widely used image classification datasets. The results of the experiments are presented in Table 2.

Table 2
Empirical results on different datasets. For each dataset, we have meticulously documented the epoch counts necessary to attain target levels of accuracy across a spectrum of methodologies (97% for MNIST, 89% for SVHN, 80% for FashionMNIST, and 45% for CIFAR-10), alongside the respective peak accuracy metrics realized (in % with the format "accuracy ± standard error"). For the GenFed approach, the subscripts enclosed in parentheses signify the diverse strategies of aggregation count modulation, corresponding to Eqs. (5) through (9). (Similar markers are used in the latter experiments.).

	MNIST		SVHN		FashionMNIST		CIFAR-10	
	ComR	Best (%)	ComR	Best (%)	ComR	Best (%)	ComR	Best (%)
GenFed (1)	250	98.640 ± 0.027	170	**91.828** ± 0.013	140	90.680 ± 0.295	470	**54.298** ± 0.319
GenFed (2)	90	98.656 ± 0.015	50	91.101 ± 0.009	30	91.212 ± 0.102	200	53.645 ± 0.043
GenFed (3)	90	98.656 ± 0.011	50	91.026 ± 0.051	30	91.217 ± 0.118	150	53.918 ± 0.148
GenFed (4)	90	**98.667** ± 0.016	50	91.157 ± 0.084	30	**91.270** ± 0.046	150	53.999 ± 0.489
GenFed (5)	140	98.270 ± 0.208	50	91.202 ± 0.033	30	91.185 ± 0.106	220	54.127 ± 0.165
FedAvg	1350	97.491 ± 0.053	1710	89.409 ± 0.021	1070	83.512 ± 0.130	2350	49.939 ± 0.401
GenFedUmf (1)	50	**98.948** ± 0.014	40	92.433 ± 0.071	30	92.282 ± 0.065	100	55.950 ± 0.215
GenFedUmf (2)	40	98.889 ± 0.023	20	92.231 ± 0.186	20	92.249 ± 0.016	40	**56.595** ± 0.099
GenFedUmf (3)	40	98.878 ± 0.105	30	**92.445** ± 0.101	30	91.197 ± 0.101	40	56.185 ± 0.212
GenFedUmf (4)	40	98.883 ± 0.027	30	92.164 ± 0.085	20	**92.307** ± 0.119	40	56.512 ± 0.023
GenFedUmf (5)	40	98.899 ± 0.013	20	92.256 ± 0.030	30	91.218 ± 0.208	40	56.151 ± 0.514
FedUmf	280	98.778 ± 0.104	320	91.072 ± 0.052	160	89.499 ± 0.151	450	55.180 ± 0.140
GenFedAU (1)	80	98.556 ± 0.107	50	91.411 ± 0.105	40	91.449 ± 0.051	100	54.595 ± 0.223
GenFedAU (2)	70	98.575 ± 0.054	50	91.129 ± 0.034	30	91.143 ± 0.019	290	51.806 ± 0.125
GenFedAU (3)	70	**98.742** ± 0.042	60	**92.894** ± 0.054	30	91.197 ± 0.101	190	55.467 ± 0.193
GedFedAU (4)	70	98.652 ± 0.016	50	92.256 ± 0.266	30	**91.639** ± 0.014	190	54.241 ± 0.061
GenFedAU (5)	70	98.606 ± 0.122	30	92.137 ± 0.137	30	91.218 ± 0.182	200	**55.801** ± 0.608
FedAU	1350	97.565 ± 0.154	1650	90.977 ± 0.205	1070	83.744 ± 0.103	2340	50.724 ± 0.032

Table 3
Comparison of time costs between various methods to achieve target accuracy. The values in the table represent the time cost required by each method relative to the time cost of FedAvg, where 1.00x corresponds to the time cost for FedAvg. In the table, our three methods adopt the strategy (3) among the five ρ_t variation strategies, which demonstrates superior performance and training speed. This strategy is given by Eq. (7).

	MNIST	SVHN	Fashion MNIST	CIFAR-10
FedAvg	1.00x	1.00x	1.00x	1.00x
GenFed	0.07x	0.03x	0.03x	0.07x
FedUmf	0.21x	0.19x	0.15x	0.20x
GenFedUmf	**0.03x**	**0.02x**	**0.02x**	**0.02x**
FedAU	1.00x	0.97x	1.01x	1.01x
GenFedAU	0.05x	0.04x	0.03x	0.08x

Experimental results demonstrate that the final model accuracy achieved through training showed substantial improvement. On the CIFAR-10 dataset, the GenFed framework delivered notable performance enhancements. The best-performing GenFed variant achieved nearly a 5% accuracy increase over the original FedAvg algorithm, and even the least effective GenFed variant surpassed FedAvg by over 2%. With optimal variations of the parameter ρ_t, the GenFedAU algorithm achieved at least a 5% accuracy gain over the original FedAU. Furthermore, our GenFedUmf algorithm showed a modest but consistent improvement compared to FedUmf. On the MNIST dataset, both GenFed and GenUmf, enhanced by our genetic strategy, attained accuracy levels exceeding 98%, outperforming the baseline FedAvg and FedUmf algorithms. Similarly, on the SVHN dataset and the FashionMNIST dataset, all methods leveraging the GenFed framework achieved accuracy levels above 90%, substantially outperforming the three baseline algorithms. This overall increase in accuracy can be attributed to the genetic mechanism's selective aggregation process, which effectively prioritizes high-quality client updates, thereby fostering a more precise global model.

Additionally, the training process exhibited substantial gains in efficiency, reducing the number of global rounds required to reach target performance levels. Specifically, in the case of the MNIST dataset, GenFed achieved 97% accuracy with only 1/15 of the training cycles necessary for FedAvg. On the SVHN dataset, even the least effective implementation of the GenFed method requires only $\frac{1}{10}$ of the training rounds needed by the FedAvg method to achieve 89% accuracy. The performance improvements of GenFedUmf over FedUmf, and GenFedAU over FedAU, follow a comparable pattern. On the FashionMNIST dataset, GenFed further demonstrated its efficiency by requiring merely 1/35 of the training cycles typically used by FedAvg to achieve an 80% accuracy, while maintaining comparable precision. Similarly, on the CIFAR-10 dataset, GenFed attained 45% accuracy using only 1/10 of the training rounds needed by FedAvg. The acceleration in training demonstrated by the GenFedAU algorithm relative to the FedAU algorithm mirrors the improvement seen between GenFed and FedAvg. While the GenFedUmf algorithm showed a more moderate increase in training speed compared to FedUmf, it nonetheless achieved measurable gains. By incorporating genetic mechanisms, this framework effectively reached the desired accuracy thresholds with significantly fewer communication rounds, illustrating its practical advantage in reducing training time and resource consumption.

Communication Overhead and Total Communication Volume Analysis: Although our method yields substantial improvements in both training efficiency and model accuracy, it does not reduce the communication overhead per round. Specifically, the GenFed framework still requires the server to broadcast the current global model to all participating clients at the start of each training round. Subsequently, each client must transmit its updated model parameters back to the server after completing local training. As a result, the communication overhead per round in our method remains identical to that of the three baseline methods used in the experiments, with a complexity of $\mathcal{O}(2\mathcal{K}\mathcal{M})$, where $\mathcal{K}$ represents the number of participating clients per round and $\mathcal{M}$ denotes the size of the model parameters. However, from the perspective of total communication volume, the GenFed framework minimizes the total communication required to achieve the target training performance. As illustrated in Fig. 3, an interesting observation is that the total communication cost for independent training of species variants within the GenFed framework is lower than the communication cost incurred by a single round of training with FedAvg to achieve the same target accuracy. This highlights the superior communication efficiency of our method, particularly when considering the total communication cost across multiple rounds.

Computational Complexity and Training Time Analysis: Although our method does not offer advantages in terms of computational complexity per round for both clients and the server, the use of genetic strategies significantly accelerates the training process. As a result, our method achieves the target training performance with fewer computational resources and less training time. The results presented in Table 3 provide strong evidence supporting this claim. Additionally, we observe the following:

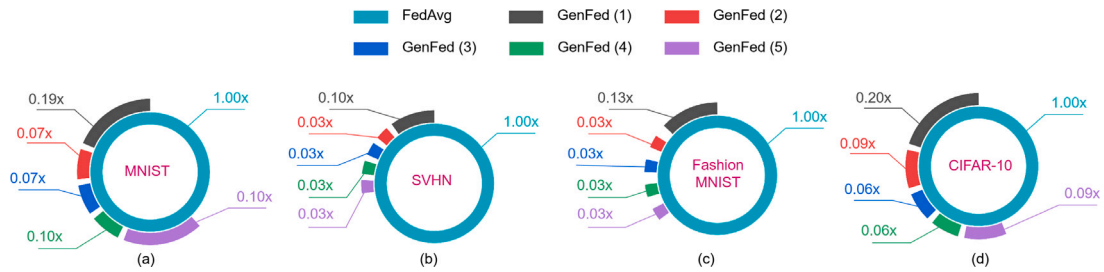

Fig. 3. Comparison of communication costs between GenFed and FedAvg across different datasets to achieve target accuracy. The central angle of each arc represents the proportion of communication cost required by each method relative to FedAvg, where 360° corresponds to one unit of communication cost for FedAvg. Subfigures (a) to (d) illustrate the communication costs required by GenFed and FedAvg to achieve target accuracies across different datasets: (a) 97% for MNIST, (b) 89% for SVHN, (c) 80% for FashionMNIST, and (d) 45% for CIFAR-10. The results highlight the communication efficiency of GenFed compared to the traditional FedAvg method.

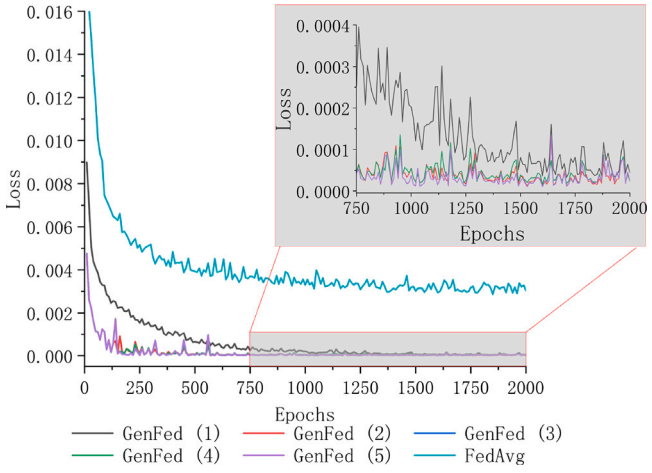

Fig. 4. Loss Curves on the SVHN Dataset. The loss curves for GenFed and FedAvg on the SVHN dataset are presented, illustrating the convergence behavior throughout the training process. To facilitate a more detailed analysis of the loss variations in the later stages of training, specifically between epochs 750 and 2000, we have zoomed in on this range and presented the corresponding curves separately.

- Compared to methods that do not employ genetic strategies, such as baseline methods, GenFed, GenFedUmf, and GenFedAU require significantly less training time.
- Our GenFed method accelerates the training process more effectively than the FedAvg method. Furthermore, when applying our genetic strategy to the FedUmf method, the training process is considerably accelerated. Notably, the resulting GenFedUmf method requires less training time than the GenFed method itself.

This analysis demonstrates the efficiency of our approach in reducing both computational cost and training time while achieving the desired model performance.

Overall, these results highlight the effectiveness and robustness of our genetic mechanism within the federated learning framework. By both enhancing model accuracy and reducing training time, our approach offers a balanced solution that meets key performance metrics critical to federated learning applications.

4.2.2. Loss curve analysis

In this section, we present the loss curves of the GenFed algorithm on the SVHN dataset and compare them with those of the traditional FedAvg method to assess their convergence behavior during training. The loss curves, shown in Fig. 4, clearly illustrate the performance differences between the two methods at various stages of training.

As depicted in Fig. 4, GenFed exhibits significantly faster convergence than FedAvg, particularly in the early stages of training. More specifically, in the later stages, a noticeable gap in final loss values emerges between the two methods. GenFed maintains a faster rate of loss reduction and ultimately achieves a lower final loss than FedAvg, demonstrating that GenFed not only accelerates convergence during prolonged training but also sustains superior performance as training progresses.

Regarding stability, FedAvg's loss curve shows considerable fluctuations, particularly in the early and mid-training phases. In contrast, GenFed's loss curve is markedly smoother, with a more consistent rate of loss reduction, particularly in the latter stages of training, where it outperforms FedAvg in both speed and stability.

Additionally, for different ρ settings, the dynamically adjusted GenFed method exhibits a significantly faster rate of loss reduction in the early stages compared to the static GenFed (1). This suggests that the dynamic mechanism enables GenFed to more effectively adapt to the evolving conditions during training, thereby improving overall training efficiency. On the other hand, GenFed (5) demonstrates relatively poor stability, with a loss curve that fluctuates significantly, particularly in the middle and later stages of training.

Overall, the analysis of the loss curves reinforces that GenFed substantially outperforms FedAvg in terms of both convergence speed and final performance, particularly excelling in both the early and late stages of training.

4.2.3. Evaluation under various attack scenarios

To evaluate the effectiveness of the GenFed method in the presence of malicious clients, we conduct attack experiments using the CIFAR-10 dataset. The specific implementations of the malicious clients are outlined as follows:

- **Label Flipping** (LF) (Jebreel, Domingo-Ferrer, Sánchez, & Blanco-Justicia, 2024): The Byzantine clients alter the labels of the local training data, flipping the labels 0 through 9 such that label x is changed to $9 - x$.
- **Mimic** (Karimireddy, He and Jaggi, 2020): The Byzantine clients send the server a copy of a (random) normal client's gradient, thereby mimicking the behavior of a benign client.
- **Inner Product Manipulation Attack** (IPM) (Xie, Koyejo, & Gupta, 2020): The Byzantine clients submit the negative vector of the mean gradient of the normal clients to the server, thereby introducing a disruptive influence on the model update process.

The results of the conducted attack experiments are comprehensively summarized in Table 4. Across all three attack scenarios, the GenFed framework exhibited significantly less performance degradation compared to FedAvg, underscoring its resilience under adversarial conditions.

Under the LF attack scenario, both FedAvg and GenFed suffered the greatest performance declines. FedAvg's accuracy dropped by over 5 percentage points; however, the most affected GenFed variant, GenFed(3), maintained an accuracy exceeding 50%, surpassing the performance of FedAvg even in non-adversarial conditions. Notably, GenFed(5), the most robust variant, displayed a counterintuitive performance improvement under the LF attack scenario.

Table 4
Experimental results for different attack methods on CIFAR-10. The percentage of accuracy decline is specified within parentheses.

	None (%)	LF (%)	IPM (%)	Minic (%)
GenFed (1)	54.298	52.373 (−3.546)	53.975 (−0.595)	53.529 (−1.416)
GenFed (2)	53.645	53.231 (−0.771)	**53.607 (−0.071)**	**54.478 (−0.311)**
GenFed (3)	53.918	52.295 (−3.010)	53.502 (−0.771)	53.175 (−1.378)
GenFed (4)	53.999	51.861 (−3.959)	53.201 (−1.478)	53.151 (−1.570)
GenFed (5)	54.127	**54.638 (0.944)**	53.889 (−0.441)	53.880 (−0.456)
FedAvg	49.939	46.995 (−5.896)	48.477 (−2.927)	47.960 (−3.963)

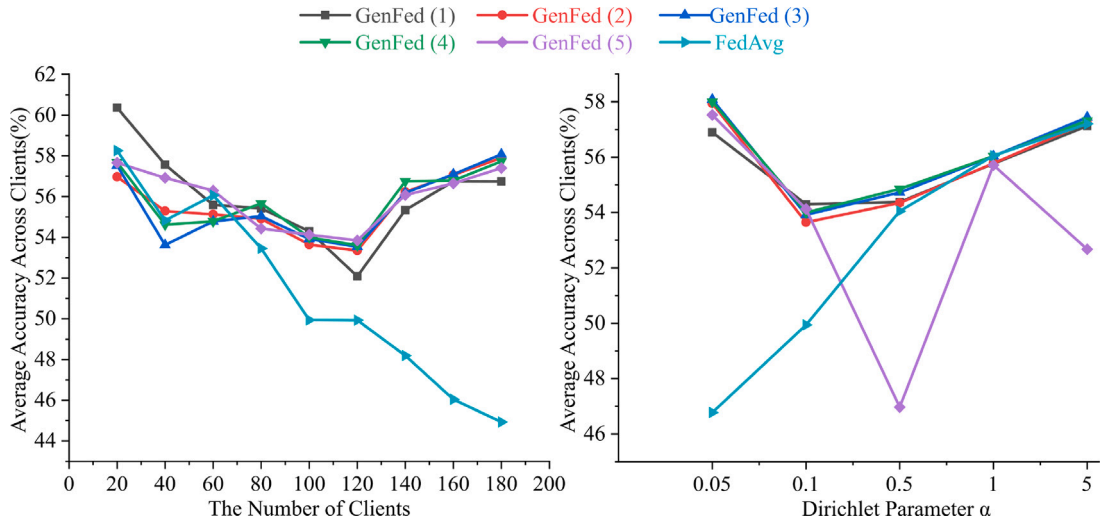

Fig. 5. Experimental results for client count and Dirichlet distribution parameters. (a) Performance trends of different algorithms with varying client counts under a Dirichlet parameter $\alpha = 0.1$. (b) Performance trends of different algorithms with a fixed client count of 100 under varying Dirichlet parameters α.

Table 5
Experimental results for varying numbers of clients. The table presents the average accuracy (in %) of the final models on the clients' test datasets, obtained after training with various methods under the corresponding number of clients.

	20	40	60	80	100	120	140	160	180
GenFed (1)	**60.367**	**57.570**	55.591	55.425	**54.298**	52.087	55.337	56.747	56.740
GenFed (2)	56.970	55.293	55.127	54.911	53.645	53.349	56.238	57.065	57.899
GenFed (3)	57.528	53.623	54.783	55.036	53.918	53.536	56.189	**57.090**	**58.077**
GenFed (4)	57.682	54.621	54.783	**55.665**	53.999	53.600	**56.754**	56.788	57.729
GenFed (5)	57.659	56.917	**56.296**	54.430	54.127	**53.852**	56.082	56.649	57.401
FedAvg	58.258	54.827	56.063	53.457	49.939	49.932	48.195	46.043	44.931

In the IPM attack scenario, FedAvg demonstrated the least performance decline among the three attack types, with a reduction nearing 3%. Despite this minimal decline, FedAvg's degradation remained more pronounced than that of most GenFed variants across other scenarios, further emphasizing GenFed's robustness under varying adversarial conditions.

Under the Minic attack scenario, GenFed achieved a reduction in performance decline exceeding 50% compared to FedAvg. This substantial mitigation highlights the adaptability of GenFed's genetic mechanism in countering specific adversarial threats, underscoring its notable advantage in preserving model integrity across diverse attack vectors.

Collectively, these findings highlight the superior robustness endowed by the GenFed framework, especially in the face of adversarial challenges, thereby reinforcing its efficacy in maintaining stable model performance under attack conditions.

4.2.4. Impact of client count and Dirichlet distribution parameters on performance

To validate the stability and superiority of our GenFed algorithm, we conducted a series of experiments that targeted various client counts and diverse heterogeneous scenarios. The results depicted in Fig. 5 effectively support this assertion.

Specifically, Table 5 presents the findings of the experiments conducted with different numbers of clients. The data clearly indicate that our GenFed method consistently outperforms the traditional approach across all tested scenarios. The advantages of our GenFed approach over the FedAvg algorithm become particularly pronounced when the number of clients reaches 60 or more. Importantly, as the client count surpasses 140, GenFed maintains a stable and relatively high level of performance. In contrast, the performance of the traditional method exhibits a marked decline, continuing to deteriorate with the increasing number of clients. This disparity underscores the robustness and scalability of GenFed in environments with a large client base, where conventional methods struggle to sustain efficiency.

Table 6 presents the experimental results across various heterogeneous scenarios. Overall, the GenFed method exhibits superior performance relative to the classical approach, particularly in settings with higher data heterogeneity, as indicated by lower α values. When data heterogeneity is moderate, the performance gap between GenFed and the traditional FedAvg narrows; however, GenFed consistently surpasses FedAvg across all settings. Interestingly, GenFed's performance under conditions of extremely high heterogeneity exceeds that observed in scenarios with only slightly lower heterogeneity, as exemplified by the results at $\alpha = 0.05$. However, it is worth noting that using a periodic ρ_t variation scheme following a complete cycle of π leads to significant fluctuations in performance under dynamic heterogeneous conditions.

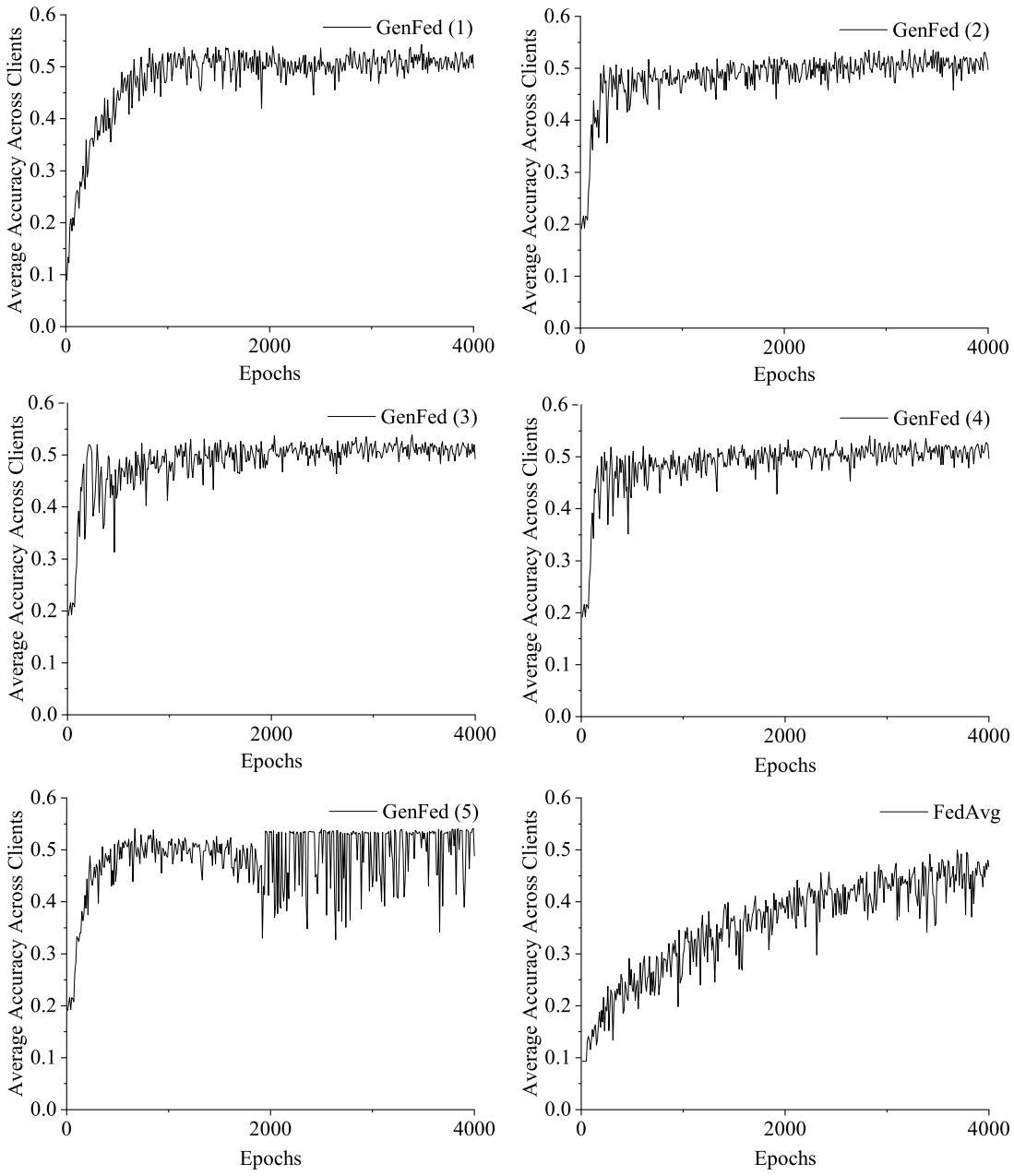

Fig. 6. The accuracy trends for each method on the CIFAR-10 dataset.

Such variability in ρ_t suggests that this scheme may be less suitable for practical applications.

4.2.5. *Influence of different ρ adjustment methods*

The results presented in Table 2 clearly demonstrate that the GenFed method achieves a significantly faster convergence rate compared to traditional algorithms, as it attains high accuracy with substantially fewer training rounds. Among the different variants, the adaptive GenFed models with varying ρ_t parameters — namely GenFed (2), GenFed (3), GenFed (4), and GenFed (5) — exhibit superior performance in accelerating the training process and achieving higher final model accuracy compared to the fixed ρ_t approach in GenFed (1).

Fig. 6 provides a detailed visualization of the accuracy trends for each method on the CIFAR-10 dataset over the course of training, further corroborating the efficacy of the GenFed methods in accelerating training. However, it is important to note that the GenFed (5) variant exhibits considerable instability in the later stages of training, which compromises its overall reliability. This observation further clarifies the instability shown in Fig. 5(b) for GenFed (5) across various heterogeneous scenarios, reinforcing its limitations for practical application.

In our experiments, we investigated five parameter adjustment strategies to identify one that combines both stability and performance.

Table 6
Experimental results under different dirichlet parameters. The table displays the average recall rates (in %) of the final models on the clients' test datasets under diverse Dirichlet parameter settings.

	0.05	0.1	0.5	1	5
GenFed (1)	56.898	**54.298**	54.381	55.761	57.125
GenFed (2)	57.939	53.645	54.350	55.778	57.249
GenFed (3)	**58.081**	53.918	54.724	56.037	**57.439**
GenFed (4)	57.999	53.999	**54.852**	56.037	57.320
GenFed (5)	57.523	54.127	46.970	55.711	52.671
FedAvg	46.777	49.939	54.048	**56.053**	57.202

Through these experiments, we found that the GenFed (5) strategy, which adjusts the number of models aggregated each round based on a triangular function with a period of π, exhibited the worst stability. Although this strategy outperformed several baseline methods in terms of accuracy, we do not recommend using it due to its instability. In contrast, other parameter adjustment strategies, whether in scenarios with a fixed 100 clients and $\alpha = 0.01$ or in those with varying client numbers or α parameters, demonstrated superior stability and performance beyond the baseline.

We hypothesize that the instability observed in GenFed (5) is due to its parameter adjustment strategy. This strategy leads to a situation in the later stages of training where each new model is heavily influenced by the single best-performing model received from the server. This optimal model dependency causes fluctuations in the training process. If such fluctuations occur early in the training, they can help the model quickly explore a broader parameter space, which benefits the overall training. However, when these fluctuations appear in the later stages, they lead to higher instability in the final model, particularly in different experimental scenarios.

Considering both the model stability and the performance of the various parameter adjustment strategies in our experiments, we recommend using either GenFed (3), which adjusts parameters linearly, or GenFed (4), which adjusts parameters based on a $\frac{\pi}{2}$-period triangular function. These two strategies combine rapid convergence with stable performance. In the earlier stages of training, fewer optimal models are aggregated to form the new round's model. As training progresses, the number of aggregated models gradually increases until it reaches the maximum value ρ_{max}. This adjustment allows for faster aggregation of a well-performing model early in training, while maintaining optimal performance and stability in the later stages.

5. Conclusion

In this study, we present an innovative framework, GenFed, developed to address the critical challenges encountered by traditional federated learning (FL) methodologies. These challenges encompass data privacy, model robustness, and the imperative for efficiency amidst data heterogeneity, computational constraints, and communication bottlenecks.

GenFed confronts these issues by integrating genetic mechanisms that enhance model aggregation strategies. This integration not only improves the efficacy of the aggregation process but also optimizes resource utilization among participating clients. Consequently, GenFed significantly accelerates convergence speed, enabling models to attain optimal performance more swiftly while maintaining resilience against adversarial attacks, which pose an increasing threat in distributed learning environments.

Furthermore, the GenFed framework demonstrates particular advantage in real-world applications characterized by a substantial diversity of client devices. In such contexts, where the heterogeneity of data and the varying computational capabilities of clients can present significant challenges, GenFed exhibits enhanced utility and adaptability. This underscores its superior applicability in large-scale distributed learning environments, where traditional methods often struggle to sustain both efficiency and effectiveness. By adeptly addressing these challenges, GenFed not only advances the field of federated learning but also paves the way for its application in complex real-world scenarios.

Looking ahead, while this study has explored the variation of the parameter ρ_t, the current strategies employed remain relatively simplistic and generic. Future research could focus on developing more refined and contextually appropriate variation strategies that can better accommodate diverse scenarios. This would enhance the flexibility and adaptability of GenFed, allowing it to perform optimally across a broader range of applications.

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.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 62276218, U2468207 and 62176221 and 52472333), Natural Science Foundation of Sichuan Province, China (NO. 2024NSFSC0490), the Fundamental Research Funds for the Central Universities, China (NO. 2682024ZTPY055), Science and Technology R and D Program of China Railway Group Limited, China (No. 2023-Major-23), Sichuan Science and Technology Program, China (Nos. 2024NSFTD0036, 2024ZHCG0166 and 2022YFG0031).

Data availability

Data will be made available on request.

References

Al-Huthaifi, R., Li, T., Al-Huda, Z., Huang, W., Luo, Z., & Xie, P. (2024). FedGODE: Secure traffic flow prediction based on federated learning and graph ordinary differential equation networks. *Knowledge-Based Systems*, [ISSN: 0950-7051] *299*, Article 112029.

Alam, T., Qamar, S., Dixit, A., & Benaida, M. (2020). Genetic algorithm: Reviews, implementations, and applications. *International Journal of Engineering Pedagogy (IJEP)*.

Albadr, M. A., Tiun, S., Ayob, M., & Al-Dhief, F. (2020). Genetic algorithm based on natural selection theory for optimization problems. *Symmetry*, *12*(11), 1758.

Ali Khowaja, S., Dev, K., Muhammad Anwar, S., & George Linguraru, M. (2025). SelfFed: Self-supervised federated learning for data heterogeneity and label scarcity in medical images. *Expert Systems with Applications*, [ISSN: 0957-4174] *261*, Article 125493.

Cheng, Z., Huang, X., Wu, P., & Yuan, K. (2024). Momentum benefits non-iid federated learning simply and provably. In *The twelfth international conference on learning representations*.

Cong, Y., Zeng, Y., Qiu, J., Fang, Z., Zhang, L., Cheng, D., et al. (2024). FedGA: A greedy approach to enhance federated learning with non-iid data. *Knowledge-Based Systems*, [ISSN: 0950-7051] *301*, Article 112201.

Damiani, C., Rodina, Y., & Decherchi, S. (2024). A hybrid federated kernel regularized least squares algorithm. *Knowledge-Based Systems*, Article 112600.

De Falco, I., Della Cioppa, A., Koutny, T., Ubl, M., Krcma, M., Scafuri, U., et al. (2023). A federated learning-inspired evolutionary algorithm: Application to glucose prediction. *Sensors*, *23*(6), 2957.

Deng, W., Xu, J., Song, Y., & Zhao, H. (2021). Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem. *Applied Soft Computing*, *100*, Article 106724.

Gao, L., Fu, H., Li, L., Chen, Y., Xu, M., & Xu, C.-Z. (2022). Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10112–10121).

Guendouzi, S. B., Ouchani, S., & Malki, M. (2022). Aggregation using genetic algorithms for federated learning in industrial cyber-physical systems. In *2022 international conference on iNnovations in intelligent sysTems and applications* (pp. 1–6). IEEE.

Hamer, J., Mohri, M., & Suresh, A. T. (2020). Fedboost: A communication-efficient algorithm for federated learning. In *International conference on machine learning* (pp. 3973–3983). PMLR.

Hsu, T.-M. H., Qi, H., & Brown, M. (2019). Measuring the effects of non-identical data distribution for federated visual classification. arXiv:1909.06335. URL https://arxiv.org/abs/1909.06335.

Huba, D., Nguyen, J., Malik, K., Zhu, R., Rabbat, M., Yousefpour, A., et al. (2022). Papaya: Practical, private, and scalable federated learning. *Vol. 4*, In *Proceedings of machine learning and systems* (pp. 814–832).

Imteaj, A., & Amini, M. H. (2022). Leveraging asynchronous federated learning to predict customers financial distress. *Intelligent Systems with Applications*, *14*, Article 200064.

Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., et al. (2019). Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. *Vol. 33*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 590–597).

Jebreel, N. M., Domingo-Ferrer, J., Sánchez, D., & Blanco-Justicia, A. (2024). LFighter: Defending against the label-flipping attack in federated learning. *Neural Networks*, *170*, 111–126.

Jiang, Y., Lu, X., Mao, W., & Lin, Y. (2023). Integrating staleness and Shapley value consistency for efficient K-asynchronous federated learning. In *2023 IEEE international conference on big data* (pp. 680–689). http://dx.doi.org/10.1109/BigData59044.2023.10386972.

Kang, D., & Ahn, C. W. (2023). Ga approach to optimize training client set in federated learning. *IEEE Access*.

Karimireddy, S. P., He, L., & Jaggi, M. (2020). Byzantine-robust learning on heterogeneous datasets via bucketing. In *The tenth international conference on learning representations*.

Karimireddy, S. P., He, L., & Jaggi, M. (2024). Byzantine-robust learning on heterogeneous datasets via bucketing. In *The twelfth international conference on learning representations*.

Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S., Stich, S., & Suresh, A. T. (2020). Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning* (pp. 5132–5143). PMLR.

Khatua, S., Mukherjee, A., & De, D. (2024). FedGen: Federated learning-based green edge computing for optimal route selection using genetic algorithm in internet of vehicular things. *Vehicular Communications*, Article 100812.

LeCun, Y. (2010). The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

LeCun, Y., Jie, F., Bourdev, L., Ranzato, M. L., Hwang, S., & Makhzani, A. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS*.

Lee, J., Ko, H., & Pack, S. (2021). Adaptive deadline determination for mobile device selection in federated learning. *IEEE Transactions on Vehicular Technology*, *71*(3), 3367–3371.

Li, B., Chen, S., & Yu, K. (2022). Model fusion from unauthorized clients in federated learning. *Mathematics*, *10*(20), 3751.

Li, Q., He, B., & Song, D. (2021). Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10713–10722).

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

Liang, Z., Zhao, K., Liang, G., Wu, Y., & Guo, J. (2024). ACFL: Communication-efficient adversarial contrastive federated learning for medical image segmentation. *Knowledge-Based Systems*, [ISSN: 0950-7051] *304*, Article 112516.

Liu, J., Jia, J., Che, T., Huo, C., Ren, J., Zhou, Y., et al. (2024). Fedasmu: Efficient asynchronous federated learning with dynamic staleness-aware model update. *Vol. 38*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 13900–13908).

Liu, J., Wang, J. H., Rong, C., Xu, Y., Yu, T., & Wang, J. (2021). Fedpa: An adaptively partial model aggregation strategy in federated learning. *Computer Networks*, *199*, Article 108468.

Luo, B., Li, X., Wang, S., Huang, J., & Tassiulas, L. (2021). Cost-effective federated learning design. In *IEEE INFOCOM 2021-IEEE conference on computer communications* (pp. 1–10). IEEE.

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics* (pp. 1273–1282). PMLR.

Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, *4*(1), 1–32.

Moreno-Álvarez, S., Paoletti, M. E., Sanchez-Fernandez, A. J., Rico-Gallego, J. A., Han, L., & Haut, J. M. (2024). Federated learning meets remote sensing. *Expert Systems with Applications*, [ISSN: 0957-4174] *255*, Article 124583.

Murata, T., Niwa, K., Fukami, T., & Tyou, I. (2024). Simple minimax optimal Byzantine robust algorithm for nonconvex objectives with uniform gradient heterogeneity. In *The twelfth international conference on learning representations*.

Nguyen, J., Malik, K., Zhan, H., Yousefpour, A., Rabbat, M., Malek, M., et al. (2022). Federated learning with buffered asynchronous aggregation. In *International conference on artificial intelligence and statistics* (pp. 3581–3607). PMLR.

Pillutla, K., Kakade, S. M., & Harchaoui, Z. (2022). Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, *70*, 1142–1154.

Qi, P., Chiaro, D., Guzzo, A., Ianni, M., Fortino, G., & Piccialli, F. (2023). Model aggregation techniques in federated learning: A comprehensive survey. *Future Generation Computer Systems*.

Rahimi, M. M., Bhatti, H. I., Park, Y., Kousar, H., & Moon, J. (2024). EvoFed: Leveraging evolutionary strategies for communication-efficient federated learning. *Vol. 36*, In *Advances in neural information processing systems* (pp. 62428–62441). Curran Associates, Inc..

Shah, P. D., & Bichkar, R. S. (2021). Secret data modification based image steganography technique using genetic algorithm having a flexible chromosome structure. *Engineering Science and Technology, an International Journal*, *24*(3), 782–794.

So, J., Güler, B., & Avestimehr, A. S. (2020). Byzantine-resilient secure federated learning. *IEEE Journal on Selected Areas in Communications*, *39*(7), 2168–2181.

Song, Y., Xie, Y., Zhang, H., Liang, Y., Ye, X., Yang, A., et al. (2021). Federated learning application on telecommunication-joint healthcare recommendation. In *2021 IEEE 21st international conference on communication technology* (pp. 1443–1448). IEEE.

Sultana, A., Haque, M. M., Chen, L., Xu, F., & Yuan, X. (2022). Eiffel: Efficient and fair scheduling in adaptive federated learning. *IEEE Transactions on Parallel and Distributed Systems*, *33*(12), 4282–4294.

Sun, Y., Shao, J., Mao, Y., & Zhang, J. (2022). Asynchronous semi-decentralized federated edge learning for heterogeneous clients. In *ICC 2022-IEEE international conference on communications* (pp. 5196–5201). IEEE.

Umbarkar, A. J., & Sheth, P. D. (2015). Crossover operators in genetic algorithms: a review.. *ICTACT Journal on Soft Computing*, *6*(1).

Wang, S., & Ji, M. (2024). A lightweight method for tackling unknown participation statistics in federated averaging. In *The twelfth international conference on learning representations*.

Wu, X., & Wang, C.-L. (2022). KAFL: achieving high training efficiency for fast-k asynchronous federated learning. In *2022 IEEE 42nd international conference on distributed computing systems* (pp. 873–883). IEEE.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747.

Xie, C., Koyejo, O., & Gupta, I. (2020). Fall of empires: Breaking byzantine-tolerant sgd by inner product manipulation. In *Uncertainty in artificial intelligence* (pp. 261–270). PMLR.

Xu, C., Qu, Y., Xiang, Y., & Gao, L. (2023). Asynchronous federated learning on heterogeneous devices: A survey. *Computer Science Review*, *50*, Article 100595.

Yang, Q., Liu, Y., Chen, T., & Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *10*(2), 1–19.

Yin, Q., Feng, Z., Li, X., Chen, S., Wu, H., & Han, G. (2024). Tackling data-heterogeneity variations in federated learning via adaptive aggregate weights. *Knowledge-Based Systems*, *304*, Article 112484.

Zang, Y., Xue, Z., Ou, S., Chu, L., Du, J., & Long, Y. (2024). Efficient asynchronous federated learning with prospective momentum aggregation and fine-grained correction. *Vol. 38*, In *Proceedings of the AAAI conference on artificial intelligence* (pp. 16642–16650).

Zhang, T., Gao, L., Lee, S., Zhang, M., & Avestimehr, S. (2023). Timelyfl: Heterogeneity-aware asynchronous federated learning with adaptive partial training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5064–5073).

Zhang, S., Zhao, Z., Liu, D., Cao, Y., Tang, H., & You, S. (2025). Edge-assisted U-shaped split federated learning with privacy-preserving for internet of things. *Expert Systems with Applications*, [ISSN: 0957-4174] *262*, Article 125494.