Knowledge-Based Systems 343 (2026) 116019

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

FedCED: Consensus enhancement in decentralized federated learning via distillation

Bin Liu [a,1], Changle Li [a,1], Qi Chu [a], Banghao Zhai [a], Zeyu Ji [a,*], Keqin Li [b,2]

[a] College of Computer Science, Northwest A&F University, Yangling, Shaanxi 712100, China
[b] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Keywords:
Federated learning
Knowledge distillation
Ring structure
Enhanced consensus

ABSTRACT

Mainstream Federated Learning (FL) methods rely on a centralized architecture, which introduces security risks like single points of failure. Decentralized architectures mitigate these risks but struggle with storing and managing historical states due to the lack of a central server. This leads to information dispersion between iterations, making it difficult for clients to reach high-quality consensus, thus reducing convergence speed and model performance, especially in ring topologies. To address these challenges, we propose the FedCED framework for ring decentralized FL, enhancing cross-client consensus capabilities. Firstly, FedCED introduces an information retention approach to integrate diverse knowledge sources and mitigating iterative fragmentation. Secondly, FedCED injects noise to promote mutual learning among nodes, eliminating isolation and strengthening client consensus. Finally, FedCED implements a historical information capture mechanism to continuously reuse old knowledge, ensuring optimization continuity. Extensive experiments demonstrate that FedCED significantly outperforms existing methods in ring topologies, achieving up to a 4.11% improvement in test accuracy, thereby validating its effectiveness.

1. Introduction

Deep learning's success fundamentally relies on centralized access to large-scale datasets, yet real-world deployment faces two irreducible constraints: inherently fragmented data ownership across institutions and legally mandated privacy protections [1]. These dual barriers render conventional data aggregation approaches both operationally impractical and legally non-compliant. Federated learning (FL) has consequently emerged as a paradigm-shifting framework that reconciles model performance with regulatory requirements, allowing decentralized entities to enhance artificial intelligence capabilities while preserving data sovereignty.

Based on the communication architecture, FL is mainly divided into centralized and decentralized types. Current mainstream research predominantly adopts a centralized architecture (as shown in Fig. 1(a)). While this approach is highly effective, it is prone to single-point failures and other security risks, which can compromise the system's robustness and scalability [2,3]. As a promising alternative, the decentralized architecture eliminates the dependency on centralized components, thereby avoiding these issues.

Decentralized FL primarily employs two types of topologies: fully connected (as shown in Fig. 1(b)) and ring structures (as shown in Fig. 1(c)). The fully connected topology, with its high degree of interconnectivity, accelerates the consensus of the model and facilitates faster convergence in non-independent and identically distributed (Non-IID) data environments. This structure also exhibits excellent fault tolerance, as the failure of individual or a few nodes typically does not impact the communication of the remaining network, ensuring stable learning. Conversely, the ring topology excels in security robustness. Its limited communication radius helps mitigate the spread of malicious models during data poisoning attacks, as malicious nodes can only directly affect their immediate neighbors. It would take several iterations for such an attack to potentially impact the entire network, providing a crucial window for detecting and countering the threat, thereby reducing the risk of widespread contamination. This paper focuses on the ring structure to explore pathways for achieving efficient and robust collaborative learning in privacy-sensitive and heterogeneous network environments.

Most existing methods use centralized designs and do not work well in decentralized settings. Decentralized FL eliminates the central server,

* Corresponding author.
E-mail addresses: liubin0929@nwsuaf.edu.cn (B. Liu), 2023056092@nwafu.edu.cn (C. Li), 13679200222@nwafu.edu.cn (Q. Chu), bhz@nwafu.edu.cn (B. Zhai), zeyu.ji@nwafu.edu.cn (Z. Ji), lik@newpaltz.edu (K. Li).
[1] These authors contributed equally to this work.
[2] IEEE Fellow.

https://doi.org/10.1016/j.knosys.2026.116019
Received 13 January 2026; Received in revised form 21 March 2026; Accepted 13 April 2026
Available online 16 April 2026

resulting in the absence of a unified node for storing and managing the global model state. This makes it difficult to continuously maintain critical information throughout the training process. This problem is especially pronounced in a ring topology: when different clients are randomly selected to participate in each training iteration, model updates can only be transmitted hop-by-hop along the ring. As a result, it becomes challenging to effectively correlate and reuse model states and progress across different iterations. Clients cannot sustain cumulative learning, making it difficult to reach consensus on a unified model state. This ultimately hampers the model's convergence speed and overall performance.

This paper introduces FedCED, a framework that tackles the model consensus challenge in ring-based decentralized FL caused by the absence of global state maintenance. It consists of three core mechanisms: firstly, FedCED introduces an information retention method that traverses models in a ring topology and employs an end-to-end client-adaptive mechanism. By preserving the training results of each node and preventing subsequent results from overwriting previous information, this approach enhances the system's ability to integrate data effectively. Secondly, FedCED incorporates a relay-enhanced model collaboration mechanism. During information retention, local models simultaneously learn from both private data and peer models from other nodes. By injecting noise, this method fosters model information interaction, reducing discrepancies among clients and promoting consensus among dispersed nodes. Finally, FedCED equips each client with a dedicated capture mechanism. At the end of each iteration, it extracts historical information from the returned models. This information is continuously reused in subsequent iterations through a noise-tuning method, supporting ongoing model optimization and convergence.

To validate the effectiveness of FedCED, extensive experiments are conducted in a decentralized environment. Experimental results show that FedCED outperforms existing representative methods by up to 4.11%, demonstrating its superior performance. The main contributions are summarized as follows:

- A information retention method is designed to integrate results from different clients within a single iteration. It ensures high-quality communication through model traversal and end-to-end client adaptation.
- A relay-enhanced model collaboration method is implemented to eliminate isolation. By injecting noise and leveraging peer models to optimize the local model, this method effectively strengthens consensus among clients.
- A information capture-based method is proposed to ensure continuous optimization. It boosts learning efficiency by locally preserving existing results and reusing historical information with synthetic data in later iterations.

The structure of this paper is arranged as follows. Section 2 reviews prior work related to this study. Section 3 provides an explanation of the preliminary work for FedCED. Section 4 introduces the proposed FedCED, including task description, objective function, and optimization algorithm. Section 5 displays experimental results on various image datasets, demonstrating that FedCED outperforms existing representative FL algorithms. Section 6 discusses the advantages and disadvantages of FedCED. Finally, Section 7 concludes this study.

2. Background and related work

This section reviews the background of this paper and the most relevant related work.

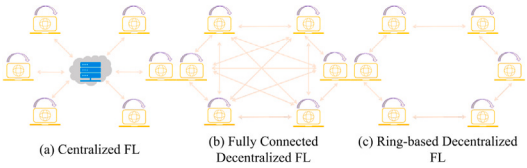


Fig. 1. Centralized and decentralized federated learning.

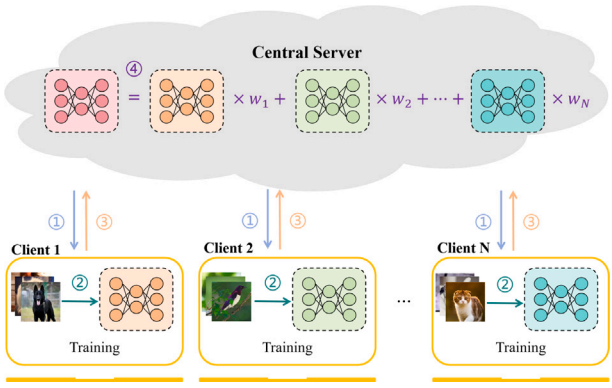


Fig. 2. The workflow of the FedAvg.

2.1. Background

FL enables collaborative machine learning while preserving data sovereignty through decentralized computation. The FedAvg algorithm [1] (visualized in Fig. 2) follows a four-phase process: (1) The central coordinator sends the global model to a subset of clients; (2) Clients perform local optimization using private datasets with multiple epochs; (3) Clients send incremental updates (parameters) back to the server; (4) The system aggregates updates through weighted averaging to form the next global model.

Building upon FedAvg, new approaches like FedProx [4], FedNova [5], FedDyn [24], FedGen [7], and FedFTG [8] have been proposed. FedProx incorporated proximal regularization terms in client-side optimization to restrict parameter divergence from the global reference. This mechanism hybridizes task-specific loss with model distance penalties in the objective function. FedNova introduces data volume awareness by quantifying client-specific data scale effects during training. During server-side aggregation, it implements adaptive aggregation coefficients proportional to local data quantities to optimize convergence dynamics. FedDyn introduced a new optimization objective in client model training. It adjusts the clients' objectives by adding linear and quadratic penalty terms, which help align the local and global models. FedGen used lightweight generation of pseudo-samples to fine-tune the global model. Building on this, FedFTG achieved adversarial learning to further enhance the distillation effect.

2.2. Related work

The related work primarily focuses on a general overview of decentralized FL, categorized based on the method of information transfer between nodes into FL methods based on knowledge distillation (KD) and those based on parameter transmission.

2.2.1. Decentralized FL based on parameter transmission

FedDgc [12] dynamically adjusts cache size to reduce redundant gradient exchanges under Non-IID data, maintaining model convergence. It enhances training efficiency in asynchronous decentralized FL by introducing delay weighting and data volume balancing. DFLGM [13] uses differential updates and a global momentum mechanism

Table 1
Comparison of various methods.

FL methods	KD	Towards Non-IID	Centralized structure	Ring structure	Fully connected structure	Personalization	Towards model heterogeneity	Noise fine-tuning
Fedavg [1]	✗	✗	✓	✗	✗	✗	✗	✗
FedProx [4]	✗	✓	✓	✗	✗	✗	✗	✗
FedNova [5]	✗	✓	✓	✗	✗	✗	✗	✗
MOON [6]	✗	✓	✓	✗	✗	✗	✗	✗
FedGen [7]	✓	✓	✓	✗	✗	✗	✗	✓
FedFTG [8]	✓	✓	✓	✗	✗	✗	✗	✓
FedNKD [9]	✓	✓	✓	✗	✗	✗	✗	✓
TARGET [10]	✓	✓	✓	✗	✗	✗	✗	✓
MHD [11]	✓	✓	✗	✗	✓	✗	✓	✗
FedDgc [12]	✗	✓	✗	✗	✗	✗	✗	✗
DFLGM [13]	✗	✓	✗	✓	✗	✗	✗	✗
Reads [14]	✗	✓	✗	✗	✓	✗	✓	✗
RingSFL [15]	✗	✓	✗	✓	✗	✗	✗	✗
DFedHP [16]	✗	✓	✗	✗	✓	✓	✓	✗
DFLStar [17]	✓	✓	✗	✗	✓	✗	✗	✗
ProFe [18]	✓	✓	✗	✗	✓	✗	✗	✗
KD-PDFL [19]	✓	✓	✗	✗	✓	✓	✓	✗
Cached-DFL [20]	✗	✓	✗	✗	✓	✗	✗	✗
DRDFL [21]	✗	✓	✗	✓	✗	✓	✗	✗
FedDCM [2]	✓	✓	✗	✓	✗	✗	✗	✗
CaPriDe [22]	✓	✓	✗	✓	✓	✓	✗	✗
BRACE [23]	✗	✓	✓	✓	✓	✗	✗	✗
FedCED (proposed)	✓	✓	✗	✓	✗	✗	✓	✓

to adaptively suppress channel noise, improving robustness. Reads facilitates personalized knowledge sharing among heterogeneous models through fine-grained layer similarity clustering and decentralized mutual learning. RingSFL [15] employs ring topology and an adaptive model splitting mechanism for efficient federated learning in heterogeneous client networks, enhancing privacy protection. DFedHP [16] dynamically generates shared parameters through a supernetwork for communication-efficient decentralized personalized FL. BRACE [23] achieves Byzantine-robust FL by utilizing 1-bit gradient quantization, consensus mechanisms, and ring aggregation. Cached-DFL [20] enhances the convergence efficiency of mobile DFL under sparse communication by using model caching relays and delay-tolerant aggregation. DRDFL [21] enables efficient FL with heterogeneous data by decoupling and collaboratively optimizing generalization and personalization modules within a ring topology.

2.2.2. Decentralized FL based on KD

FedDCM [2] introduced mutual knowledge distillation. The local privacy model effectively acquire knowledge from other clients by mutually learning from shared models. MHD [11] achieved decentralized federated learning via multi-head distillation using multiple proxy clients as auxiliary heads. CaPriDe [22] proposed an encrypted KD method for decentralized FL that does not affect training accuracy. By encrypting the model's predictions, the privacy protection capabilities of FL are further strengthened. DFLStar [17] uses self-distillation to suppress local drift and incorporates an intelligent neighbor selection mechanism to achieve communication-efficient decentralized FL. ProFe [18] integrates KD, prototype learning, and quantization techniques to enable communication-efficient decentralized FL. KD-PDFL [19] dynamically optimizes collaboration weights by assessing the similarity of client models, facilitating privacy-preserving personalized FL.

Existing research predominantly relies on centralized architectures. In complex data environments and diverse application scenarios, extensive experiments and rigorous theoretical developments have significantly advanced FL training methods. However, these methods typically depend on centralized structures, which face inherent limitations in decentralized environments — such as low-quality communication, client isolation, and the lack of unified storage — making it challenging to achieve cross-client consensus, particularly in ring decentralized architectures. In contrast to existing studies, this paper introduces the innovative FL framework FedCED, based on KD theory and built on a more secure ring architecture. This framework enhances the consensus among client models while ensuring data sovereignty, thereby optimizing prediction accuracy. Table 1 clearly highlights the differences and unique aspects of this study compared to previous research.

3. Preliminaries

In this section, the relevant preliminaries will be introduced, including federated learning and knowledge distillation.

3.1. Federated learning

3.1.1. Notion

In this work, assume there are N clients, denoted as $C_1, C_2, \ldots, C_N$. The dataset $D_k = \{(x_i^k, y_i^k)\}_{i=1}^{N_k}$ is stored separately in C_k. $p_k(x, y)$ and $|D_k|$ represent the data distribution and size of D_k, respectively.

3.1.2. Problem statement

Generally speaking, federated learning optimizes the following objective over $D_G \overset{\Delta}{=} \cup_{i\in[N]} D_i$:

$$\arg\min_{\omega} \sum_{k=1}^{N} w_k \sum_{i=1}^{N_k} \mathcal{L}(x_i^k, y_i^k, \omega), \tag{1}$$

where ω represents the model in machine learning, $\mathcal{L}$ is the loss function that measures training error, and w_k is calculated as follows:

$$w_k = |D_k|/|D_G|. \tag{2}$$

To prevent privacy leakage, the datasets of clients are not allowed to be exchanged.

3.1.3. Non-IID setting

In FL, client datasets often exhibit Non-IID patterns, where discrepancies in joint probability distributions $p_m(x, y)$ and $p_n(x, y)$ arise. These can be expressed as $p_m(x|y)p_m(y)$ or $p_m(y|x)p_m(x)$, categorizing them into different Non-IID types. Label distribution disparity, where client-specific label distributions $p_m(y)$ vary while class-conditional feature distributions $p_m(x|y)$ remain constant, is the most prominent challenge. For example, in autonomous driving, some institutions may focus on specific road conditions and therefore have more related data.

Table 2
The definition of notations.

N	Number of clients
M	Number of nodes
C_i	The ith client
N_i	The ith node
ω_i	The model stored in N_i
$\omega_{i,j}$	The new model obtained by training ω_i in N_j
ω_g	A capturer
ω_s	Student model in fine-tuning
D_i	The private dataset stored in N_i
D	All samples participating in this FL iteration.
w_i	The proportion of $\|D_i\|$ in $\|D\|$
P_i	The set of models initialized in N_i
$\mathcal{L}_{CE}(\cdot)$	Cross-entropy function
t	Client participation iterations
ϵ	Random data that follow a normal distribution
$p_n(y)$	Distribution of noise data labels
$\mathcal{D}_{KL}(\cdot)$	KL divergence entropy function
$S(\cdot)$	Softmax function
$\exp(\cdot)$	Exponential function

3.2. Knowledge distillation

Deep neural networks typically employ high-parameter architectures to process raw input patterns, necessitating significant computational overhead and substantial memory footprint. This inherent structural characteristic creates deployment barriers for edge computing scenarios with hardware limitations. Structural optimization strategies like model compression address this challenge through parameter-space reduction. Among model compression methodologies, KD has emerged as a predominant technique.

In KD, the student model ω_S adopts a streamlined architectural configuration with reduced parametric dimensionality, whereas the teacher network ω_T demonstrates sophisticated structural complexity with enhanced parametric capacity. Knowledge transfer occurs as ω_S learns from ω_T's feature activations and output distributions within a shared input space D_h, aided by dynamic parameter synchronization [2]. Typically, ω_T's probabilistic guidance serves as supervisory signals, where soft probability distributions replace categorical labels during ω_S's optimization. Kullback–Leibler (KL) divergence is used to quantify predictive alignment between the two networks.

$$\mathcal{L}_{KD} = \mathcal{D}_{KL}(S(\omega_S(D_h)) \parallel S(\omega_T(D_h))), \tag{3}$$

where $S(\cdot)$ represents the softmax function, and the calculation of $\mathcal{D}_{KL}(\cdot)$ is as follows:

$$\mathcal{D}_{KL}(P \parallel Q) = \sum_i P_i \cdot \log(P_i/Q_i), \tag{4}$$

where P and Q represent two different probability distributions.

By learning from soft labels, the simple student model can acquire strong generalization capabilities based on the complex teacher model. However, the inherent representational gap between teacher–student architectures in KD can lead to inferential degradation, as student networks struggle to fully assimilate the teacher's knowledge.

4. Methodology

This section introduces FedCED, the proposed FL method designed to address the limitations of the ring architecture with targeted solutions. In each iteration, FedCED randomly selects a group of clients as ring nodes, where each node shares its model with adjacent nodes according to the topology to ensure efficient communication. To enhance model performance, local models are augmented using data from relay nodes, and consensus with peer models is strengthened by injecting noise, thereby mitigating client discrepancies. After knowledge transfer, local nodes use a capture mechanism to extract knowledge from the returned models and store it locally, enabling continuous optimization across iterations. The notions covered in this section are shown in Table 2. The corresponding algorithms are summarized in Algorithms 1, 2, and 3 (see Fig. 3).

Algorithm 1: Information Retention

Input:
$\boldsymbol{M}$: Number of nodes;$\{\boldsymbol{N_i}\}_{i\in\{0,\dots,M-1\}}$: Nodes in a decentralized structure; $\{\boldsymbol{\omega_i}\}_{i\in\{0,\dots,M-1\}}$; $\{\boldsymbol{D_i}\}_{i\in\{0,\dots,M-1\}}$;
Output:
$\{\boldsymbol{P_i}\}_{i\in\{0,\dots,M-1\}}$: The returned sets of models;
for *i from 0 to* $M-1$ **do**
  $P_i \leftarrow \emptyset$;
  $P_i \leftarrow P_i + \{\omega_i\}$;
for *i from 0 to* $M-1$ **do**
  for *j from 0 to* $M-1$ **do**
    if $i \neq 0$ **then**
      N_j receives P from $N_{(j-1+M)\%M}$;
      $\omega \leftarrow P\{0\}$;
      Train ω using D_j to obtain ω'. The training method follows Algorithm 2;
      $P \leftarrow P + \{\omega\}$;
    if $i \neq M-1$ **then**
      N_j sends P to $N_{(j+1)\%M}$;
return $\{P_i\}_{i\in\{0,\dots,M-1\}}$

Algorithm 2: Relay-Enhanced Model Collaboration

Input:
$\boldsymbol{E_t}$: Number of local training epochs;
$\boldsymbol{P}$: The model set received;
$\boldsymbol{N_j}$: A relay node in decentralized FL;
$\boldsymbol{D_j}$: The private dataset on N_j;
$\boldsymbol{\omega_j}$: A model stored on N_j. $\boldsymbol{\eta}$; $\boldsymbol{t}$; $\boldsymbol{\lambda_M}$;
Output:
$\boldsymbol{P}$: The model set;
$\omega \leftarrow P\{0\}$;
for *e from 1 to* E_t **do**
  $(X, y, p_n(y)) \leftarrow \omega_g(\epsilon \sim \mathcal{N}(0,1))$;
  for $\{X_D, y_D\}$ *from* D_j **do**
    $\mathcal{L}_M \leftarrow \mathcal{D}_{KL}(S(\omega(X)) || S(\omega_j(X))$;
    $\mu \leftarrow \lambda_M \times t$;
    $\mathcal{L} \leftarrow \mathcal{L}_{CE}(\omega_i(X_D)), y_D) + \mu \times \mathcal{L}_M$;
    $\omega' \leftarrow \omega - \eta \times \bigtriangledown \mathcal{L}$;
$P \leftarrow P + \{\omega'\}$;
return P

4.1. Information retention method

In [25], two ring-structured FL paradigms are summarized: the continuous paradigm and the aggregation paradigm. In the continuous paradigm, clients directly learn from the predecessor model, while in the aggregation paradigm, multiple models are aggregated before training. Both paradigms can lead to subsequent information overwriting previous results, thereby weakening global consensus. To address this issue, FedCED is designed with information retention.

At the beginning of the FL, the system randomly selects M clients as nodes in a decentralized structure, represented as $N_0, N_1, N_2, \dots, N_{M-1}$. The nodes are arranged in a ring topology, where, for instance, N_0 is connected to both N_1 and N_{M-1}, and N_1 is connected to N_2 and N_0. At the same time, N_i stores the local model ω_i and the private dataset D_i.

For a given node N_i, once model transfer begins, it generates an empty set, denoted as:

$$P_i \leftarrow \emptyset, \tag{5}$$

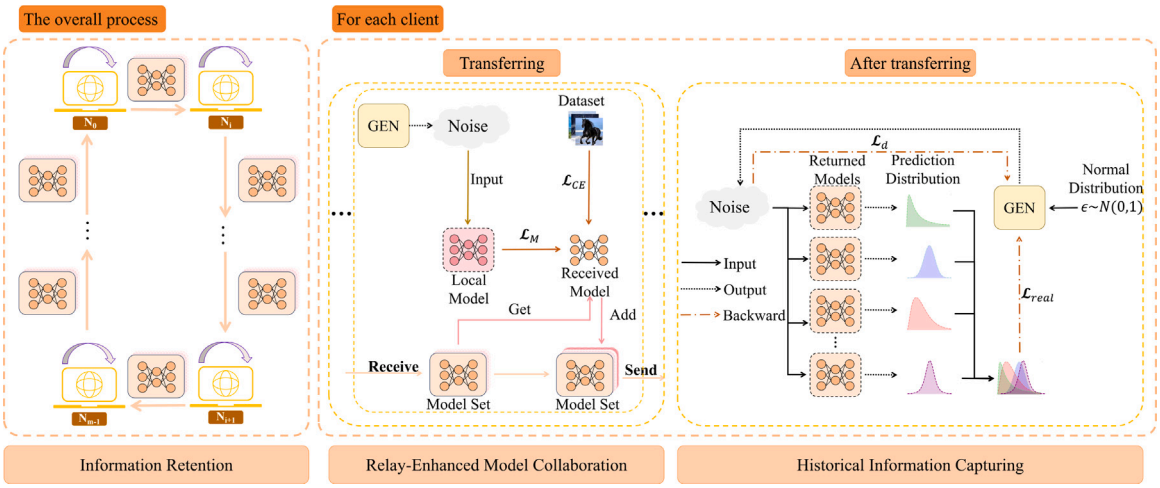


Fig. 3. The overall architecture of FedCED.

Algorithm 3: Historical Information Capturing

Input:
E_g: The number of epochs for noise generator updating;
E_{KD}: The number of epochs for noise fine-tuning;
$\{\omega_{i,j}\}_{j\in\{0,\ldots,M-1\}}$: The model set returned to node N_i;
$\{|D_i|\}_{i\in\{0,\ldots,M-1\}}$: Amount of samples of each node; M; ω_g; $|D|$;
η; λ_d;
Output:
ω_g, ω_i: The noise generator and the local model;
$\mathcal{L}_{real} \leftarrow 0$;
for *e from 1 to* E_g **do**
  $(X, y, p_n(y)) \leftarrow \omega_g(\epsilon \sim \mathcal{N}(0,1))$;
  for *j from 0 to* $M-1$ **do**
    $w_j \leftarrow |D_j|/|D|$;
    $\mathcal{L}_{real} \leftarrow \mathcal{L}_{real} + w_j \times \mathcal{L}_{CE}(\omega_{i,j}(X), y)$;
  $\mathcal{L}_d \leftarrow \exp\left(-\frac{1}{|\epsilon|}\sum_{i,j\in\{1,2,\ldots,|\epsilon|\}}^{i\neq j}(||\omega_g(\epsilon_i) - \omega_g(\epsilon_j)||_2\right)$;
  $\mathcal{L} \leftarrow \mathcal{L}_{real} + \lambda_d \times \mathcal{L}_d$;
  $\omega_g \leftarrow \omega_g - \eta \times \nabla\mathcal{L}$;
return ω_g

and adds ω_i to the set:

$$P_i = P_i + \{\omega_i\}. \tag{6}$$

After completing the above operations, P_i is passed to the adjacent node $N_{(i+1)\%M}$. On $N_{(i+1)\%M}$, ω_i is obtained from P_i, and it is used to train $D_{(i+1)\%M}$, resulting in $\omega_{i,(i+1)\%M}$. Then, $\omega_{i,(i+1)\%M}$ is also added to the set P_i:

$$P_i = P_i + \{\omega_{i,(i+1)\%M}\}. \tag{7}$$

Next, $N_{(i+1)\%M}$ passes P_i to $N_{(i+2)\%M}$, and uses ω_i to train the private dataset $D_{(i+2)\%M}$, resulting in $\omega_{i,(i+2)\%M}$. After the training is completed, $\omega_{i,(i+2)\%M}$ is added to P_i using the method in Eq. (7). It is worth noting that at each relay node, every newly generated model is derived from ω_i.

The above process is repeated continuously until P_i returns to N_i from $N_{(i-1+M)\%M}$. Finally, at N_i, ω_i is used to train the local dataset D_i, resulting in $\omega_{i,i}$.

After the model transfer is complete, N_i will obtain the model set$\{\omega_{i,j}\}_{j\in\{0,\ldots,M-1\}}$. During the iteration process, each node simultaneously performs the same operations as N_i, ultimately obtaining a model set. The final model parameters can be obtained by performing a weighted average of the model set.

4.2. Relay-enhanced model collaboration method

In Section 4.1, the local model ω_i is transmitted via P_i to the relay node N_j for training on its private dataset D_j. This provides an opportunity for ω_i and the relay node model ω_j to "communicate". Common methods, such as contrastive learning, utilize supervised training to bring ω_i closer to ω_j. However, since the performance of ω_j on the overall dataset is unmeasurable, it cannot be assumed that ω_j performs better than ω_i. If ω_j performs worse than ω_i, using this approach may lead to a decline in accuracy, thereby affecting the performance of FL.

Another common method for extracting information from models is KD. To enhance the effectiveness of transfer, FedCED uses a data-free knowledge distillation (DFKD) approach. Specifically, FedCED deploys an information capturer for each client, generating noise in N_j to extract information from ω_j, thereby enhancing the performance of ω_i. At the end of each iteration, the generator extracts historical data from the returned models.

First, N_j generates a random dataset ϵ following a standard normal distribution:

$$\epsilon \sim \mathcal{N}(0,1). \tag{8}$$

Next, ϵ is input into the capturer to produce noise:

$$X = \omega_g(\epsilon, y), \tag{9}$$

where X represents the noise data, y represents the category labels of the noise data, and ω_g represents the capturer.

In KD, the noise data (X, y) generated by the relay node's capturer can serve as a shared dataset D_h. To facilitate this process, FedCED designs the $\mathcal{L}_M$ function to extract information from ω_j, which is computed as follows:

$$\mathcal{L}_M = \mathcal{D}_{KL}S(\omega_i(X)) \parallel S(\omega_j(X)), \tag{10}$$

where $D_{KL}(\cdot)$ is computed as shown in Eq. (4), and $S(\cdot)$ denotes the softmax function.

Additionally, ω_i must be enhanced using D_j, so its overall optimization objective is defined as follows:

$$\min_{\omega_i} \mathbb{E}_{\substack{y_D\sim D_j \\ y\sim p_n(y)}}\left[\mathcal{L}_{CE}(\omega_i(X_D), y_D) + \mu \times \mathcal{L}_M\right], \tag{11}$$

where X_D represents the samples drawn from D_j, y_D are the category labels of X_D, and μ denotes the weight of $\mathcal{L}_M$. $\mathcal{L}_{CE}$ is the cross-entropy loss, which is calculated as follows:

$$\mathcal{L}_{CE}(P, Q) = -\sum_i P_i \cdot log(Q_i), \tag{12}$$

where P and Q represent two probability distributions.

Moreover, because it is uncertain whether ω_i or ω_j performs better, directly applying KD could potentially degrade the performance of ω_i. To address this issue, FedCED employs a direct approach: it evaluates the number of times each client participates in FL and adjusts the model learning degree accordingly. Specifically, the number of times N_j participates in FL (t) influences the weight μ of $\mathcal{L}_M$, thereby controlling the extent of distillation:

$$\mu = \lambda_M \times t, \tag{13}$$

where λ_M is a constant.

By adjusting the weight of $\mathcal{L}_M$, the degree to which ω_i learns from ω_j can be controlled. This approach helps mitigate the negative impact of less efficient models and enhances the overall training effectiveness.

Finally, use Eq. (11) to update ω_i and obtain the new model, which is then incorporated into P_i. When a new node joins, the aforementioned steps are repeated.

4.3. Historical information capturing mechanism

At the end of each iteration, captures on each client extract information from all participating clients and store it locally. In future iterations, the noise produced by these captures will be used to enhance the effect of relay model augmentation.

Due to the privacy protection mechanisms inherent in FL, captures cannot access the private datasets of individual clients to acquire information. Therefore, after the information transfer is complete, utilizing the set of returned models to acquire information becomes a feasible solution.

Specifically, node N_i leverages the set of returned models $\{\omega_{i,j}\}_{j\in\{0,\dots,M-1\}}$ to acquire information. To better retain the historical information, it is necessary to make the noise distribution closer to the real data distribution. For this purpose, the $\mathcal{L}_{real}$ function is designed as follows:

$$\mathcal{L}_{real} = \sum_{j=0}^{M-1} w_j \times \mathcal{L}_{real}^{j}, \tag{14}$$

where $\mathcal{L}_{real}^{j}$ represents the loss of X on $\omega_{i,j}$, which is calculated as follows:

$$\mathcal{L}_{real}^{j} = \mathcal{L}_{CE}(S(\omega_{i,j}(X)), y), \tag{15}$$

where $S(\cdot)$ is the softmax likelihood function.

Typically, the Non-IID nature of FL data can lead to performance divergence in models trained by different clients. To better adapt the capture to the input space of different client models, FedCED introduces a weighting factor w_i to reflect the importance of each client in the loss calculation. The weight w_i quantifies the proportion of $|D_i|$ relative to $|D|$, allowing clients with richer data to dominate $\mathcal{L}_{real}$. This weighting strategy effectively integrates results from different clients, helping the generator produce noise that is closer to the real data distribution.

Additionally, to enhance the fidelity of the noise to the original data distribution across various categories, it is necessary to expand the class diversity within X. For this purpose, the $\mathcal{L}_d$ function is designed as follows:

$$\mathcal{L}_d = \exp\left(-\frac{1}{|\epsilon|^2}\sum_{i,j\in|\epsilon|}^{i\neq j} ||\omega_g(\epsilon_i) - \omega_g(\epsilon_j)||_2\right), \tag{16}$$

where ϵ_i is a vector in ϵ, $\exp(\cdot)$ denotes the exponential function, and $\|\cdot\|_2$ represents the Euclidean norm.

Finally, the overall optimization objective for the capture is as follows:

$$\min_{\omega_g} \mathbb{E}_{\substack{\epsilon\sim\mathcal{N}(0,1)\\ y\sim p_n(y)}}\left[\mathcal{L}_{real} + \lambda_d \times \mathcal{L}_d\right], \tag{17}$$

where $p_n(y)$ represents the distribution of labels for the noise and λ_d is a constant.

As training progresses, the information captured by the clients continues to grow, thereby further enhancing the effect of relay knowledge augmentation in subsequent training iterations.

4.4. Communication overhead modeling

This subsection models the communication overhead incurred by a single node over the entire training process. For simplicity, we assume that all clients share the same model architecture.

As established in Section 4.1, for P_i, each additional relay hop incurs an extra ω_i:

$$\mathrm{Comm}_{P_i} = (1 + 2 + \cdots + m) \times |\omega| = \frac{m \times (m-1)}{2}|\omega|. \tag{18}$$

From Eq. (18), the per-iteration communication overhead is:

$$\mathrm{Comm}_{\mathrm{Iter}} = \sum_{i=0}^{m-1} \mathrm{Comm}_{P_i} = \frac{m^2 \times (m-1)}{2}|\omega|. \tag{19}$$

Therefore, over the entire training horizon, FedCED's total communication overhead is:

$$\mathrm{Comm}_{\mathrm{FedCED}} = \sum_{i=1}^{T} \mathrm{Comm}_{\mathrm{Iter}} = \frac{T \times m^2 \times (m-1)}{2}|\omega|. \tag{20}$$

[25] categorizes existing decentralized FL methods into two main paradigms: the continual paradigm and the aggregation paradigm. In the continual paradigm, clients directly receive models from predecessor nodes and perform local learning. In the aggregation paradigm, clients first aggregate model parameters from multiple nodes before proceeding with training. Therefore, the communication costs for these two paradigms in a single iteration are as follows:

$$\mathrm{Comm}_{\mathrm{Continual\_Iter}} = m^2 \times |\omega|, \tag{21}$$

$$\mathrm{Comm}_{\mathrm{Aggregate\_Iter}} = \frac{m^2 \times (m-1)}{2}|\omega|. \tag{22}$$

Thus, the communication costs for the continual paradigm and the aggregation paradigm are as follows:

$$\begin{aligned}\mathrm{Comm}_{\mathrm{Continual}} &= \sum_{i=1}^{T} \mathrm{Comm}_{\mathrm{Continual\_Iter}} \\ &= T \times m^2 \times |\omega|,\end{aligned} \tag{23}$$

$$\begin{aligned}\mathrm{Comm}_{\mathrm{Aggregate}} &= \sum_{i=1}^{T} \mathrm{Comm}_{\mathrm{Aggregate\_Iter}} \\ &= \frac{T \times m^2 \times (m-1)}{2}|\omega|.\end{aligned} \tag{24}$$

By comparing the three, we can observe the following relationship:

$$\mathrm{Comm}_{\mathrm{FedCED}} = \mathrm{Comm}_{\mathrm{Aggregate}} > \mathrm{Comm}_{\mathrm{Continual}}. \tag{25}$$

Therefore, we can conclude that FedCED's communication cost is similar to that of the aggregation paradigm and greater than that of the continual paradigm.

5. Experiments

In this section, the effectiveness of FedCED is validated through experiments. Firstly, in Section 5.1, all implementation details are summarized. Secondly, in Section 5.2, FedCED is compared with several baseline algorithms under a decentralized structure; in Section 5.3, the performance of all FL methods is tested under different hyperparameters; and in Section 5.4, other metrics relevant to this study are evaluated. Thirdly, Section 5.5 employs ablation analysis to confirm FedCED's essential components. Finally, Section 5.6 demonstrates practical FL efficacy through tri-dataset benchmarking. Due to space constraints, certain experiments have been moved to the supplementary material, available online.

Table 3
Experimental platform.

Configuration	Value
Graphics processor units	NVIDIA Tesla T4 × 4
GPU memory	16 GB × 4
Operating system	Ubuntu 18.04.2 LTS (64-bit)
Deep learning framework	Pytorch 1.13.1
CUDA version	11.6

Table 4
Architecture of the capture module.

$z \in \mathbb{R}^d \sim \mathcal{N}(0,1)$ $m = \text{Map}(y) \in \mathbb{R}^M,\ y \in \{1,\ldots,M\}$
FC(z) → 4096 FC(m) → 4096
Concat → 8192 Reshape, BN → $128 \times 8 \times 8$ Conv2D, BN, LeakyReLU → $128 \times 8 \times 8$ Upsampling → $128 \times 16 \times 16$ Conv2D, BN, LeakyReLU → $64 \times 16 \times 16$ Upsampling → $64 \times 32 \times 32$ Conv2D, Tanh → $3 \times 32 \times 32$

5.1. Implementation details

5.1.1. Baselines

To ensure a fair comparison, the experiments are conducted in a decentralized environment, comparing FedCED with FedAvg [1], FedProx [4], FedNKD [9], FedFTG [8], FedDCM [2], CaPriDe [22], and BRACE [23]. Among these methods, FedAvg, FedProx, and BRACE are based on parameter transmission, while FedNKD, FedFTG, FedDCM, and CaPriDe are based on KD. FedAvg, FedProx, FedNKD, and FedFTG are originally designed for centralized architectures and are referred to as De-FedAvg, De-FedProx, De-FedNKD, and De-FedFTG, respectively, in this context. On the other hand, FedDCM, CaPriDe, and BRACE are designed specifically for decentralized architectures.

5.1.2. Experimental platform

The configuration of the experimental platform is shown in Table 3. The experimental platform consists of four NVIDIA Tesla T4 GPUs, each equipped with 16 GB of memory, running the Ubuntu 18.04.2 LTS (64-bit) operating system, with the Pytorch 1.13.1 deep learning framework, and CUDA version 11.6.

5.1.3. Datasets

The experiments selected the common datasets: CIFAR10 [26], MNIST [27], and Fashion-MNIST [28].

Aligned with [8,24], Dir(β) is employed to generate Non-IID client data distributions, where lower β values intensify class imbalance gradients.

In addition, real-world FL settings inherently exhibit imbalanced client data quantities. To simulate this, the experiment uses a log-normal distribution to emulate the differences in data amount across clients. The parameter α controls the variability in client data sizes; when α is small, the difference in data amount among clients is smaller, while a larger α results in greater disparity.

5.1.4. Network architecture

The experiments use the most common machine learning model: ResNet-18 [29]. Meanwhile, the capture module in FedCED adopts DFAD [30], and its architecture is shown in Table 4.

5.1.5. Hyperparameters

For all FL methods, the following hyperparameters are included in the experiment: the local training epochs E, the number of FL iterations I, the number of clients N, the proportion of active clients C, the data heterogeneity score β, the client data amount score α, the client model initial learning rate L_C, the capturer initial learning rate L_G, the learning rate decay factor per iteration η, the weight decay wd, the batch size B, λ_M, and λ_d. By default, these parameters are set as follows: $E = 5$, $I = 100$, $N = 30$, $C = 0.4$, $\beta = 0.3$, $\alpha = 0.3$, $L_C = 0.1$, $L_G = 0.01$, $\eta = 0.99$, $wd = 0.001$, $\lambda_M = 0.025$, and $\lambda_d = 1.0$. Meanwhile, for CIFAR10, $B = 100$; for MNIST, $B = 64$; and for Fashion-MNIST, $B = 200$.

5.2. Performance comparison

5.2.1. Convergence accuracy

Table 5 displays the test accuracy of all methods on the CIFAR10, MNIST, and Fashion-MNIST datasets. The experimental settings are ($N = 10, C = 0.6$), ($N = 20, C = 0.5$), and ($N = 30, C = 0.4$). All experiments were repeated three times, and the results represent the average accuracy across all clients. At the same time, the best-performing experimental results under the same conditions are highlighted in bold. Additionally, centralized training was performed under the same conditions, and the corresponding test results are reported. The results demonstrate that FedCED outperforms existing FL methods in the decentralized environment across all settings. On CIFAR10, accuracy improves by 2.34% to 4.11%; on MNIST, the improvement ranges from 2.03% to 2.25%; and on Fashion-MNIST, the improvement ranges from 2.01% to 2.85%.

Compared to other methods, FedCED shows a significant improvement in accuracy, primarily due to its effective enhancement of client consensus in ring structures. Moreover, with the exception of FedDCM, other KD-based FL methods also outperform De-FedAvg and De-FedProx in terms of performance. This indicates that in environments with high data heterogeneity, simple parameter averaging methods can easily lead to a decline in system performance. Additionally, it is observed that in certain cases, De-FedFTG can outperform CaPriDe. This is because CaPriDe uses local private datasets from clients as shared data in KD, but in highly heterogeneous environments, these local private datasets fail to effectively reflect the global knowledge distribution. Meanwhile, although the BRACE method improves consensus among nodes, its 1-bit design significantly compromises accuracy.

5.2.2. Convergence efficiency

To evaluate the convergence efficiency of each method, Fig. 4 illustrates the learning curves of all FL methods over 100 iterations. Additionally, we report the number of iterations required for each FL method to reach specific target test accuracies (30% and 50% for CIFAR10; 70% and 90% for MNIST; 55% and 75% for Fashion-MNIST). Since FedDCM and BRACE failed to achieve some of the targets, its results are not included. Furthermore, using the number of iterations required by De-FedAvg to reach the target accuracy as a baseline, we calculate the speedup ratio for each FL method. As shown in Table 6, the method that achieves the target accuracy the fastest, along with its corresponding iteration count and speedup ratio, is highlighted in bold.

Combining the insights from Fig. 4 and Table 6, we observe that FedCED experiences a slower accuracy growth in the early stages of training, allowing other methods to initially surpass it. However, FedCED's accuracy growth subsequently accelerates significantly, eventually surpassing all other methods by a substantial margin. Although De-FedAvg, De-FedNKD, and De-FedFTG reach the target accuracies with fewer iterations in the early stages, their final test accuracies remain lower than FedCED's. This phenomenon occurs because FedCED can access more private data from clients, whereas De-FedAvg, De-FedProx, De-FedNKD, and De-FedFTG are limited to accessing data from adjacent nodes due to technical constraints. In the early stages,

Table 5
Test accuracy (%) of different FL methods on CIFAR10, MNIST and Fashion-MNIST ($\beta = 0.3$).

Centralized training	CIFAR10			MNIST			Fashion-MNIST		
	85.51			99.56			92.12		
	$N = 10$ $C = 0.6$	$N = 20$ $C = 0.5$	$N = 30$ $C = 0.4$	$N = 10$ $C = 0.6$	$N = 20$ $C = 0.5$	$N = 30$ $C = 0.4$	$N = 10$ $C = 0.6$	$N = 20$ $C = 0.5$	$N = 30$ $C = 0.4$
De-FedAvg	56.48	54.31	51.30	94.71	94.54	94.21	78.53	77.64	77.48
De-FedProx	57.21	55.27	52.61	94.78	94.68	94.29	80.41	78.98	77.32
De-FedNKD	59.71	57.11	53.90	95.18	95.01	94.96	83.77	81.40	79.39
De-FedFTG	60.55	58.29	55.22	94.91	94.57	94.11	80.71	79.15	78.82
FedDCM	36.91	36.25	32.83	80.71	79.76	78.41	59.75	57.79	55.75
CaPriDe	60.26	57.63	54.75	97.01	96.57	96.14	83.92	82.67	80.46
BRACE	56.22	54.99	49.83	95.46	95.28	95.11	81.25	80.26	79.11
FedCED	**64.66**	**61.64**	**57.56**	**99.04**	**98.61**	**98.39**	**86.41**	**84.68**	**83.31**

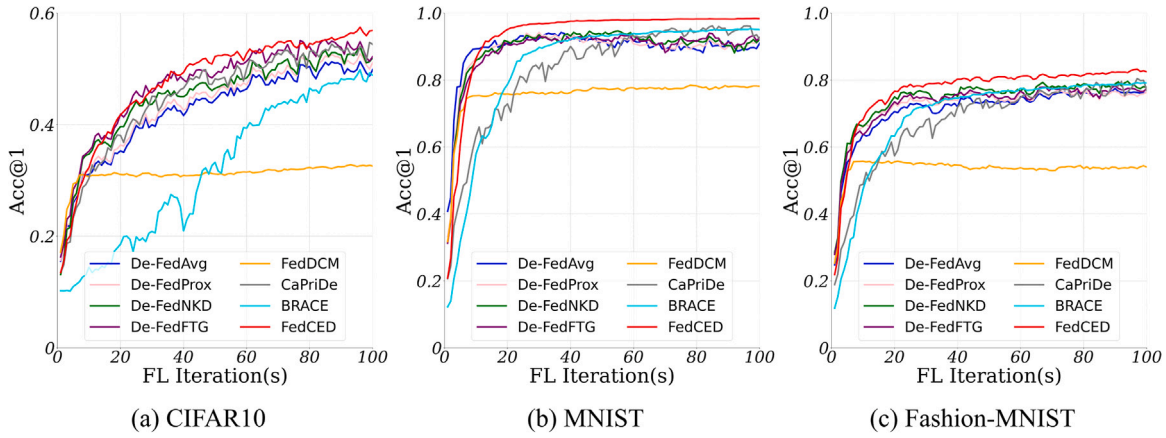


Fig. 4. Learning curves for (a) CIFAR10, (b) MNIST, and (c) Fashion-MNIST over 100 iterations ($\beta = 0.3$).

Table 6
Evaluate the convergence speed of different FL methods on CIFAR10, MNIST, and Fashion-MNIST ($N = 30, C = 0.4, \beta = 0.3$), using the number of iterations to first reach the target test accuracy as the reference metric. At the same time, the speedup of each method is calculated based on De-FedAvg.

Method	CIFAR10				MNIST				Fashion-MNIST			
	30%		50%		70%		90%		55%		75%	
	Iter.	Speed	Iter.	Speed	Iter.	Speed	Iter.	Speed	Iter.	Speed	Iter.	Speed
De-FedAvg	11	1×	77	1×	**4**	**1**×	13	1×	5.67	1×	41.67	1×
De-FedProx	8.67	1.27×	69	1.12×	4.67	0.86×	18.33	0.71×	5.67	1×	41	1.02×
De-FedNKD	7.67	1.43×	57	1.35×	5	0.8×	17.33	0.75×	**5**	**1.13**×	22.33	1.87×
De-FedFTG	**7.33**	**1.50**×	52.33	1.47×	5.33	0.75×	20.67	0.63×	5.33	1.06×	22.67	1.84×
FedDCM	7.67	1.43×	–	–	6	0.67×	–	–	7	0.81×	–	–
Capride	9.67	1.14×	61	1.26×	15.67	0.26×	37.67	0.35×	13.67	0.41×	58.33	0.71×
BRACE	42	0.26×	–	–	15.33	0.26×	32	0.41×	14.67	0.39×	42	0.99×
FedCED	8.67	1.27×	**41.67**	**1.85**×	7	0.57×	**12**	**1.09**×	5.67	1×	**18.67**	**2.23**×

these methods achieve rapid accuracy improvements due to the smaller training sample size; however, in the later stages, their performance gains are constrained by the sample size limitations.

Simultaneously, we analyzed the training stability of all methods. As illustrated in Fig. 4, FedCED exhibits the most stable learning curve compared to the other methods. The learning curves of De-FedAvg, De-FedProx, De-FedNKD, and De-FedFTG show significant fluctuations, with even oscillatory declines observed on the MNIST dataset. Although FedDCM maintains a relatively stable learning curve, its overall performance is lacking. BRACE experiences severe oscillations on CIFAR10, although it remains relatively stable on MNIST and Fashion-MNIST. CaPriDe demonstrates the lowest stability in its learning curve. The underlying reason for this phenomenon is that FedCED effectively integrates the information from all clients participating in the FL process, whereas De-FedAvg, De-FedProx, De-FedNKD, and De-FedFTG tend to gather information from neighboring nodes. As training progresses, the continuous changes in neighboring nodes lead to instability in the training process. Similarly, CaPriDe, which utilizes local private datasets for knowledge distillation, also experiences fluctuations in its learning curve.

5.3. The impact of hyperparameters

5.3.1. Data heterogeneity

Fig. 5(a) illustrates the accuracy under different β configurations. FedCED consistently outperforms all benchmark methods, underscoring its robustness across various scenarios. The figure shows that the accuracy of all methods improves as data heterogeneity decreases (i.e., as β increases). Additionally, when data heterogeneity is low, the performance differences among the methods become negligible. This is because reduced data heterogeneity minimizes client drift, resulting in a more stable convergence trajectory for the models, which in turn enhances overall accuracy.

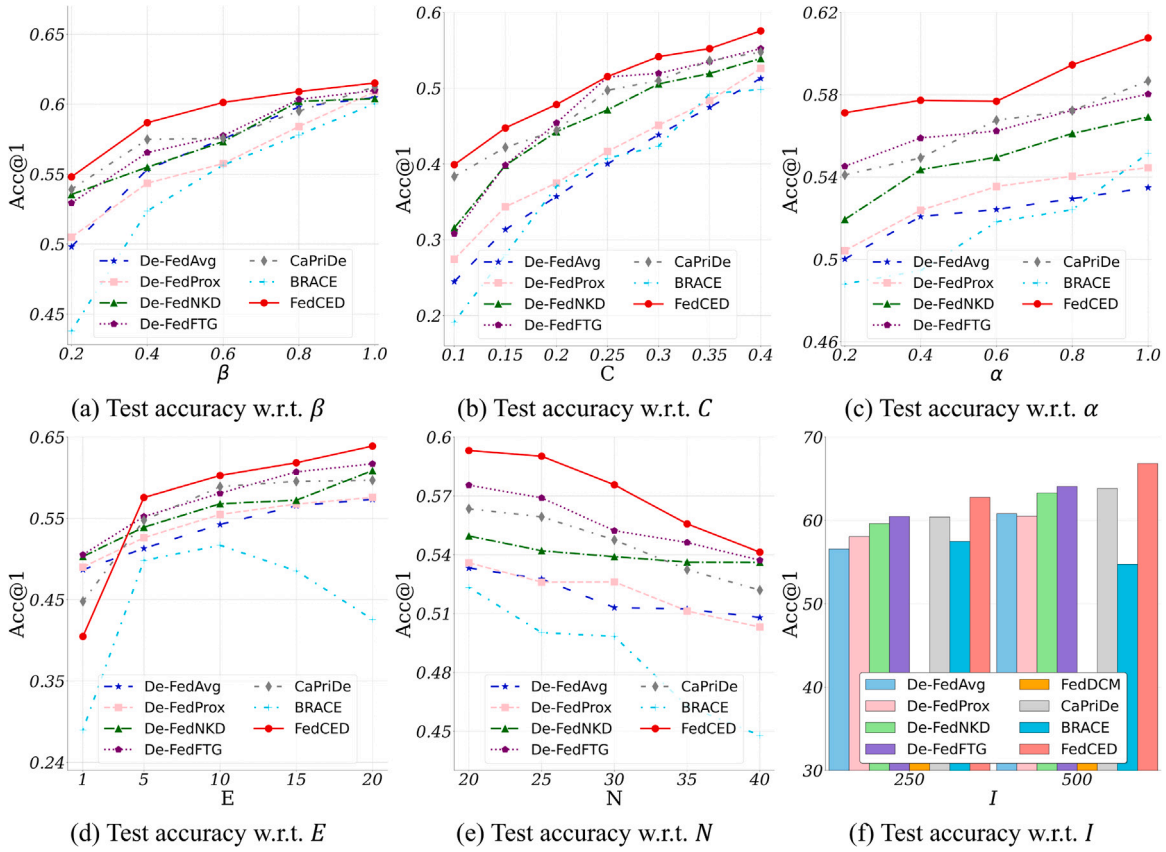


Fig. 5. The impact of hyperparameter. Test accuracy w.r.t. (a) data heterogeneity, (b) proportion of active clients, (c) client data amount, (d) local training epochs, (e) number of clients, and (f) number of iterations.

5.3.2. Proportion of active clients

Fig. 5(b) compares FL method accuracy across active client proportions (C). FedCED also achieves the best performance in this plot. In Fig. 5(b), for all FL methods, accuracy improves rapidly as more clients participate in training. At the same time, FedCED achieves significantly greater accuracy improvements when fewer clients are involved, compared to other FL methods. This is because FedCED can leverage the capturer to store information in each iteration and reuse it, which significantly enhances its performance.

5.3.3. Data amount for clients

Fig. 5(c) provides the test accuracy under different values of α. In Fig. 5(c), FedCED achieves the best performance, indicating that FedCED is efficient across various client data amount settings. Furthermore, it is observed that as α increases, the test accuracy of all FL methods improves. This is because as α increases, the disparity in the amount of data across clients grows. In this case, some clients have less data while others have more. Since FL randomly selects clients to participate in training, this gives the system the opportunity to leverage more data, thereby achieving higher accuracy.

5.3.4. Local training epoch

Fig. 5(d) reports the test accuracy of all FL methods under different local training epochs (E). FedCED achieved the best performance except for $E = 1$. As more local training epochs are used, all FL methods show faster accuracy improvements. This occurs because longer local training periods enhance client learning, leading to better FL system performance. For $E = 1$, FedCED performs poorly because with fewer local training epochs, extracting each participant knowledge is less beneficial than gathering information from adjacent nodes. The lower test accuracy of CaPriDe compared to De-FedAvg, De-FedProx, De-FedNKD, and De-FedFTG further supports this point. Additionally, since the capturer cannot effectively capture information, the produced noise significantly differs from the real situation. Fine-tuning KD with poor noise further reduces FedCED's performance.

5.3.5. The number of clients

Fig. 5(e) examines the test accuracy variation across FL methods with varying client quantities (N). FedCED maintains superior accuracy in this evaluation. The results reveal a consistent performance degradation across all methods as client participation scales up. This occurs because higher client numbers amplify data fragmentation, raising the complexity of information integration and consequently impairing generalization capability.

5.3.6. The number of iterations in FL

Fig. 5(f) reports the testing accuracy of all methods for different numbers of iterations ($I = 250$ and $I = 500$). To align with the change in the number of iterations, the learning rate decay factors for each iteration are set to $\eta = 0.995$ and $\eta = 0.998$, respectively. In Fig. 5(f), FedCED achieves the best testing performance. Additionally, when comparing Fig. 5(f) with Table 5, there is a significant increase in the testing accuracy of all methods, indicating that an increase in the number of iterations leads to a substantial performance gain for FL methods.

5.4. Other test experiments

5.4.1. Visualization of the prediction distribution

To provide a more intuitive illustration of each method's performance, 2D t-SNE visualization was applied to the experimental results on the MNIST dataset under the condition ($N = 30, C = 0.4$), as shown in Fig. 6. Fig. 6(b)–(i) visualize ensemble predictions across clients,

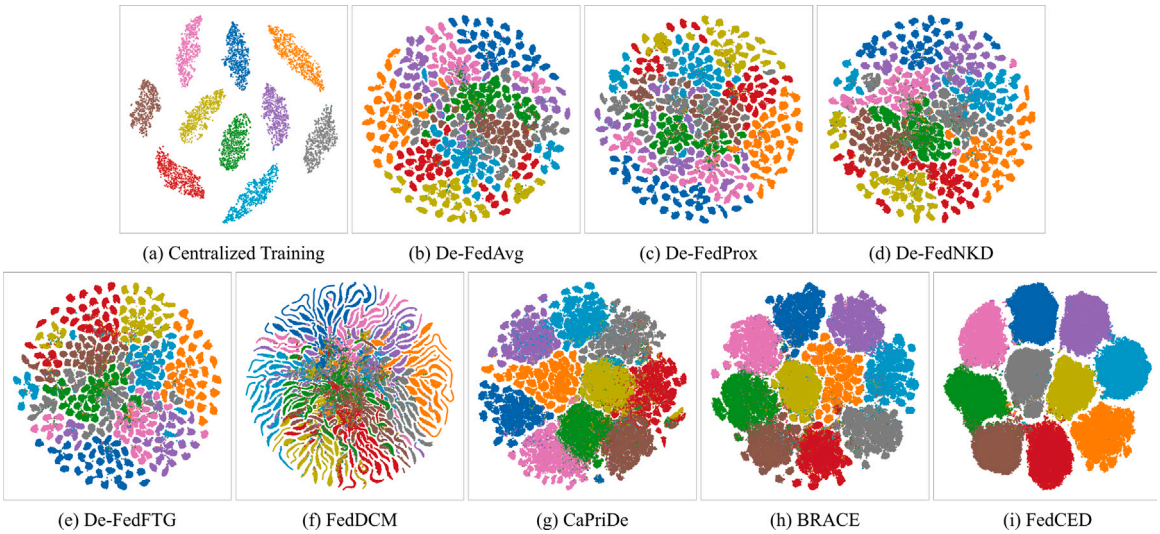


Fig. 6. Visualizing the training results using 2D t-SNE.

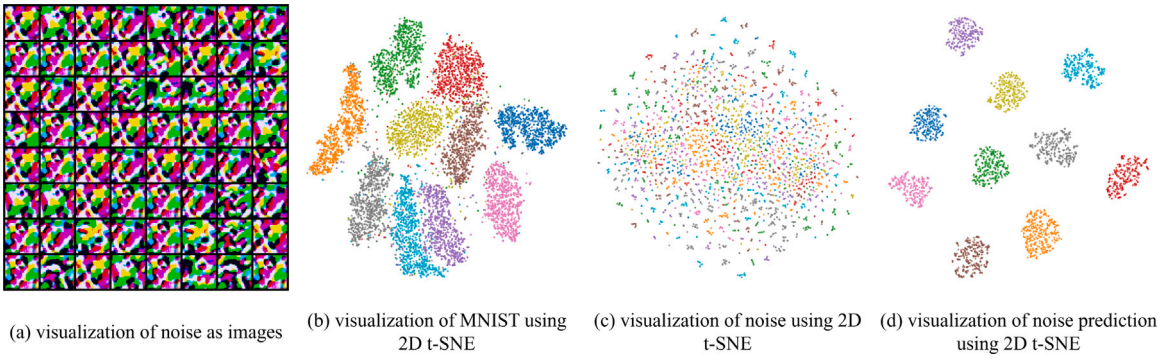


Fig. 7. Visualization of noise and noise prediction.

which clearly demonstrate how different FL methods perform on the same data distribution. This helps in understanding the performance differences among the methods.

From these two-dimensional t-SNE visualizations, it is evident that the training results of De-FedAvg, De-FedProx, De-FedNKD, De-FedFTG, and FedDCM are suboptimal. These methods exhibit blurred class boundaries and non-compact within-class distributions, indicating significant discrepancies in client predictions. Consequently, the overall distribution appears more scattered and chaotic. In contrast, the prediction distributions for CaPriDe and BRACE are more concentrated, but they still suffer from the aforementioned issues. FedCED, on the other hand, showcases the most concentrated prediction distribution, with minimal performance differences between clients and more consistent training states. This highlights FedCED's ability to more effectively coordinate the learning processes across different nodes, thereby reducing model inconsistencies and enhancing cross-client consensus.

5.4.2. Test accuracy of clients

Fig. 8 plots test accuracies across 10 clients for all methods on CIFAR-10, MNIST, and Fashion-MNIST. FedCED delivers the strongest overall client performance. For De-FedAvg, De-FedProx, De-FedNKD, De-FedFTG, CaPriDe, and BRACE, a minority of clients surpass FedCED, but the majority fall short. These methods also show pronounced cross-client disparity—some clients perform well while others perform very poorly. By contrast, client test accuracies under FedCED fluctuate only slightly, indicating stronger consensus among participants. This underscores FedCED's greater robustness and its ability to limit the impact of extreme client behavior on overall stability.

Table 7
Assess the convergence speed of IR combined with centralized FL on CIFAR10 ($N = 30, C = 0.4, \beta = 0.3$), using target accuracy attainment iterations as the evaluation metric. Meanwhile, calculate the speedup of each method based on De-FedAvg.

Method	CIFAR10			
	30%		50%	
	Iter.	Speed	Iter.	Speed
De-FedAvg+IR	11.67	0.94×	67.67	1.14×
De-FedProx+IR	11.67	0.94×	63.67	1.21×
De-FedNKD+IR	8	1.38×	39	1.97×
De-FedFTG+IR	**6.67**	**1.65×**	**28.67**	**2.69×**

5.4.3. The combination of information retention (IR) and centralized FL methods

To verify that IR can effectively transfer information in a decentralized environment, it is combined with centralized FL methods De-FedAvg, De-FedProx, De-FedNKD, and De-FedFTG, and tests are conducted.

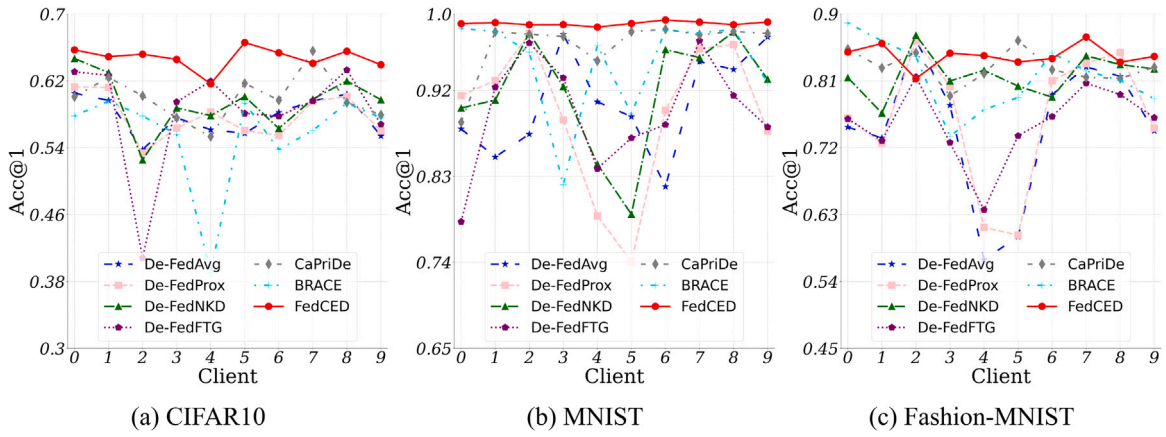
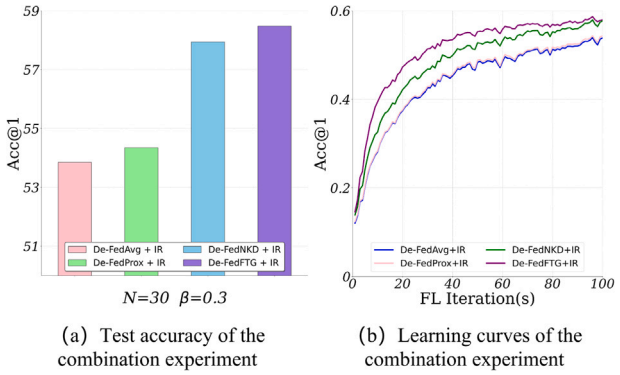


Fig. 8. Client performance on (a) CIFAR10, (b) MNIST, and (c) Fashion-MNIST.



(a) Test accuracy of the combination experiment

(b) Learning curves of the combination experiment

Fig. 9. The combination of IR and centralized FL methods experiment. (a) test accuracy and (b) learning curves.

Fig. 9(a) displays the test accuracy of De-FedAvg+IR, De-FedProx+IR, De-FedNKD+IR, and De-FedFTG+IR. De-FedFTG+IR achieves peak test accuracy in Fig. 9(a), correlating with Table 5 metrics. By comparing Fig. 9(a) with Table 5, it is observed that for all centralized FL methods, the simple incorporation of IR leads to a significant performance improvement.

Fig. 9(b) shows the learning curves of De-FedAvg+IR, De-FedProx+IR, De-FedNKD+IR, and De-FedFTG+IR. To evaluate convergence speed, Table 7 lists target accuracy attainment iterations per FL method. Additionally, Table 7 calculates the speedup of the other methods with De-FedAvg as the benchmark. The fastest-converging method's iteration count and speedup are bold in Table 7. De-FedFTG+IR demonstrates minimal iterations, aligning with Fig. 4 and Table 6 metrics. By comparing Table 7 with Table 6, it is observed that after incorporating IR, the convergence speed of these FL methods slows down in the early stages but accelerates significantly in the later stages. Meanwhile, by comparing Fig. 9(b) with Fig. 4, it is found that the training process becomes more stable after IR is combined with these centralized FL methods. This suggests that IR can both accelerate convergence speed and improve training stability.

5.4.4. Privacy-preserving capability of historical information capture

Since FedCED implements a capturer on each client to retain historical information, there is a potential risk of privacy leakage. However, based on our observations, the capturer can only capture shared features of the real data and cannot capture the unique properties of individual data points. At the same time, these shared features are high-level abstractions from the perspective of computers, which differ significantly from the real data and are incomprehensible to humans. Furthermore, the training process does not directly access or handle raw client data, thereby further mitigating potential leakage risks.

Fig. 7(a) depicts the noise's spatial distribution in image form. The visualized noise manifests as a featureless random data cluster. By comparing Fig. 7(a) with the original dataset, it is evident that there is a significant difference between the two, indicating that local client cannot obtain private data from other clients through the capturer.

Fig. 7(c) presents the visualization of the noise generated by FedCED after training on MNIST, using 2D t-SNE. In Fig. 7(c), the noise data points are mixed across categories, with no clear boundaries, which is in stark contrast to the MNIST dataset visualized using 2D t-SNE in Fig. 7(b). Thus, even under analytical scrutiny, noise fails to attain parity with genuine data characteristics.

Fig. 7(d) exhibits the noise prediction results for all clients. In Fig. 7(d), clear boundaries are observed for each class, and the class distribution closely resembles that in Fig. 7(b). This indicates that the noise encodes class-specific patterns detectable only via client-localized model inference. The local client can significantly improve performance by fine-tuning its model with noise.

Overall, based on our current assessments, FedCED does not appear to introduce additional negative impacts on data security, suggesting that its approach to capturing historical information aligns with privacy protection requirements.

5.4.5. Communication overhead

Fig. 10 summarizes the communication overhead of the FL methods on three datasets, obtained via multi-node simulation on a single GPU. Because the overhead depends solely on the volume of transmitted model parameters and is dataset-agnostic, the results are consistent across datasets. BRACE exhibits the lowest overhead, owing to its Ring-All-Reduce scheme and 1-bit quantization. FedDCM, grounded in the continual learning paradigm (Eq. (23)), likewise incurs a relatively low cost. De-FedAvg, De-FedProx, De-FedNKD, and De-FedFTG all follow the aggregation paradigm (Eq. (24)) and therefore have identical overhead. FedCED's overhead is on par with these aggregation-based methods, aligning with the theoretical prediction in Eq. (25). By contrast, CaPriDe additionally transmits distillation-related information atop the aggregation paradigm, resulting in the highest overhead.

5.4.6. GPU memory

Fig. 11 reports GPU memory usage for various FL methods on CIFAR-10, MNIST, and Fashion-MNIST. On Fashion-MNIST, the batch size is 200 (compared to 100 for the other datasets), so all methods use more GPU memory than on CIFAR-10 and MNIST. Across methods, De-FedNKD, De-FedFTG, and FedCED exhibit notably higher memory consumption, primarily because the DFKD design introduces an auxiliary generator that adds nontrivial overhead. De-FedAvg, De-FedProx,

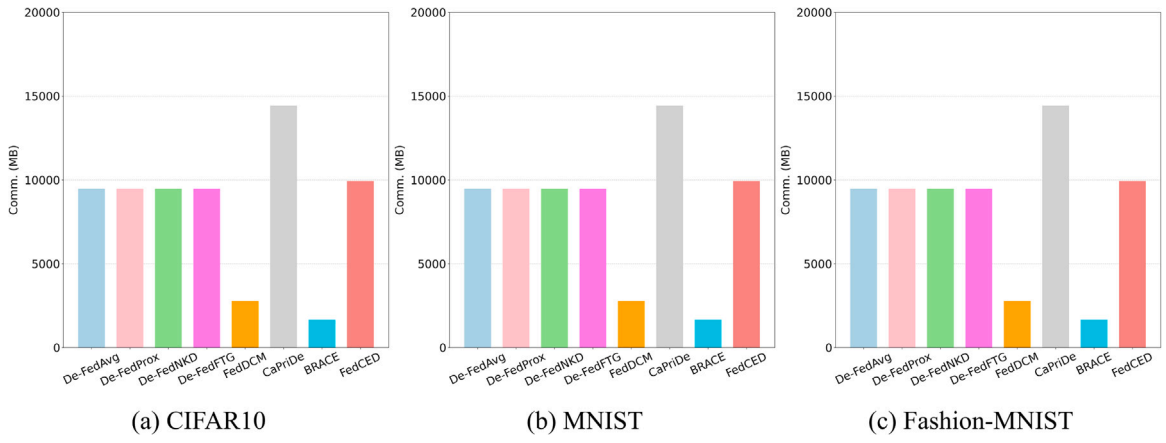


Fig. 10. Communication overhead on (a) CIFAR10, (b) MNIST, and (c) Fashion-MNIST.

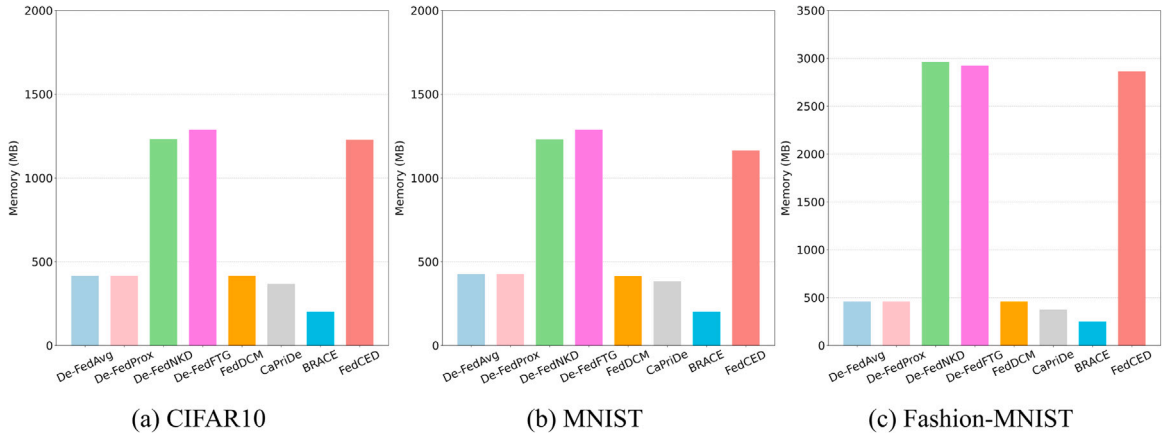


Fig. 11. GPU memory on (a) CIFAR10, (b) MNIST, and (c) Fashion-MNIST.

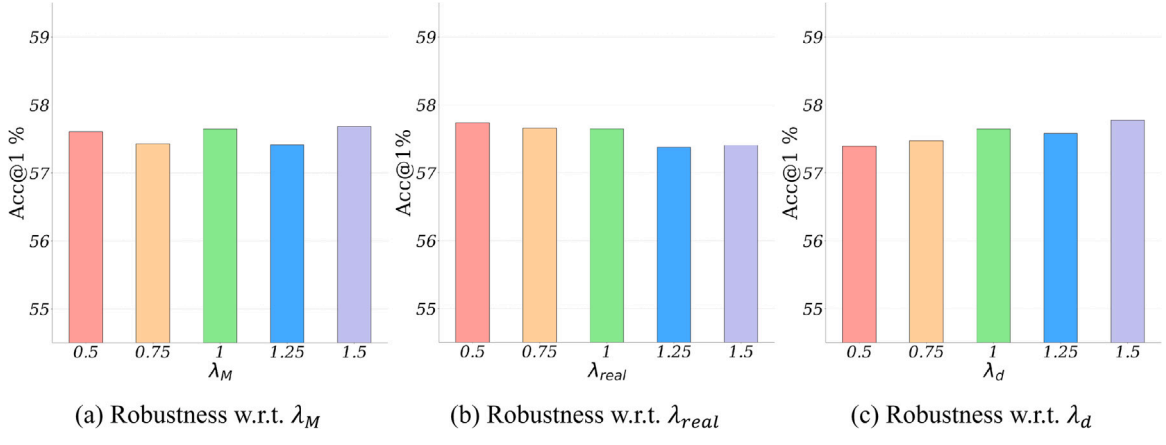


Fig. 12. Robustness tests on the loss functions $\mathcal{L}_M$, $\mathcal{L}_{real}$, and $\mathcal{L}_d$ are conducted by varying λ_M, λ_{real}, and λ_d. (a) λ_M, (b) λ_{real}, and (c) λ_d.

FedDCM, and CaPriDE have comparable memory footprints, reflecting the cost of the underlying baseline framework. BRACE leverages 1-bit quantization to further reduce memory, yielding the lowest demand.

5.5. Ablation study

5.5.1. The necessity of each module and loss

Table 8 presents the test accuracies of FedCED on the CIFAR10 dataset after 100 iterations ($\beta = 0.3$), with certain loss functions or modules omitted. In this table, IR, RMC, and HIC stand for information retention, Relay-enhanced Model Collaboration, and Historical Information Capture, respectively. The ablation study results reveal that removing any single module inevitably leads to a performance decline. Furthermore, eliminating multiple components results in even more pronounced performance degradation. Similarly, with respect to the loss functions, removing a single loss function causes a decrease in system performance, and the removal of multiple loss functions exacerbates this negative impact.

Table 8
The impact of each component in FedCED. The experiment uses the CIFAR10 dataset with $N = 30$ and $C = 0.4$.

	Method	CIFAR10
Baseline	FedCED	57.56
Loss	$-\mathcal{L}_M$	56.36
	$-\mathcal{L}_{real}$	55.13
	$-\mathcal{L}_d$	56.51
	$-\mathcal{L}_{real} - \mathcal{L}_d$	55.04
Module	−IR	52.46
	−RMC	56.36
	−HIC	55.81
	−IR&RMC	52.55
	−IR&HIC	52.02
	−RMC&HIC	55.69
	−IR&RMC&HIC	51.94

Table 9
FedCED performance under different network topologies. Experiments are conducted on CIFAR-10, MNIST, and Fashion-MNIST.

Topology	CIFAR10	MNIST	Fashion-MNIST
Ring graph (baseline)	57.73	98.39	83.31
Erdős–Rényi graph	65.45	98.86	85.04
Small-world graph	47.52	97.69	79.54
Complete graph	69.53	99.06	86.99

5.5.2. Robustness of $\mathcal{L}_M$, $\mathcal{L}_{real}$, and $\mathcal{L}_d$

To test the robustness of $\mathcal{L}_{real}$, Eq. (26) is rewritten as:

$$\min_{\omega_g} \mathbb{E}_{\substack{\epsilon\sim\mathcal{N}(0,1)\\ y\sim p_n(y)}} \left[\lambda_{real} \times \mathcal{L}_{real} + \lambda_d \times \mathcal{L}_d\right]. \tag{26}$$

This experiment will explore the robustness of $\mathcal{L}_M$, $\mathcal{L}_{real}$, and $\mathcal{L}_d$ by varying the values of λ_M, λ_{real}, and λ_d. In the experiment, λ_M was selected from [0.015, 0.02, 0.025, 0.03, 0.035], and λ_{real} and λ_d were selected from [0.5, 0.75, 1.0, 1.25, 1.5]. Fig. 12 shows the test accuracy under different hyperparameters, with all results demonstrating similar performance. Meanwhile, based on the experimental results, the variance under different λ_M, λ_{real}, and λ_d was calculated, which were 0.004, 0.001, and 0.006, respectively. This indicates that the loss function is not sensitive to the selection of hyperparameters, exhibiting good robustness.

5.5.3. Impact of topology

Although FedCED is designed for ring topologies, it can be readily adapted to other decentralized topologies by adjusting its communication protocol. To assess the impact of different topologies — including dynamic and partially connected networks — on FedCED's performance, we modified its communication logic and ran experiments. Table 9 summarizes test accuracy across three datasets under Ring, Erdős–Rényi, Small-world, and Complete graph [31] topologies. The fully connected Complete graph, with the greatest level of consensus, achieves the best performance; the structured Ring graph is stable; the randomly connected Erdős–Rényi graph performs second best; whereas the Small-world graph, which combines community structure with random shortcuts, is challenging. This is largely because its high local clustering conflicts with FedCED's sequential relay mechanism: information tends to circulate within local cycles rather than traverse shortcuts, which in turn hinders the formation of global consensus. These results show that FedCED's performance is highly contingent on the communication patterns of the underlying topology and motivate the development of adaptive, topology-aware path-planning mechanisms—a key focus of our future work.

5.6. Experiments on real-world datasets

This subsection evaluates FedCED's performance across complex real-world benchmarks — Apple [32], Caltech101 [33], and GTSRB [34].

Table 10
Test accuracy (%) on real-world datasets Apple, Caltech101 and GTSRB.

Method	Apple	Caltech101	GTSRB
De-FedAvg	58.61	30.65	80.06
De-FedProx	57.18	30.55	79.93
De-FedNKD	57.71	35.40	84.51
De-FedFTG	64.22	36.82	84.93
FedDCM	48.15	19.12	50.23
CaPriDe	58.17	34.69	83.13
BRACE	61.02	36.35	83.69
FedCED	**68.94**	**38.87**	**89.38**

Standardized experimental protocols included 8:2 training-test splits for Apple and Caltech101, while GTSRB utilized its prespecified split. All image inputs were standardized to 32×32 resolution before training. All other experimental settings remain the same as those described in Section 5.1.

Table 10 presents comparative performance metrics across three FL benchmarks. Data analysis reveals that while De-FedNKD and CaPriDe achieve accuracy gains over De-FedAvg on Caltech101 and GTSRB, they underperform on Apple's detection tasks. This limitation highlights their inadequacy for fine-grained detection scenarios. Notably, FedCED maintains consistent superiority across all tasks, demonstrating robust applicability in practical FL deployments.

6. Discussion

In this section, a theoretical analysis of model heterogeneity and computational overhead of FedCED will be presented.

6.1. Model heterogeneity

In FedCED, the local model is transferred to other clients for training and then returned through a decentralized topology. Therefore, FedCED does not require all client models to be consistent, which significantly enhances the flexibility of FedCED. Similarly, this approach is also compatible with parameter-averaging FL methods.

6.2. Computational overhead

FedCED's computational overhead mainly arises from two components: relay-enhanced model collaboration and historical information capture. Compared with existing KD-based decentralized federated learning methods (e.g., FedDCM and CaPriDe), FedCED only introduces an additional information-collection phase via the capture module. Nevertheless, the dominant overhead is attributable to relay-enhanced collaboration, while the capture module adds only a minor extra cost. Hence, although FedCED incurs some additional computation, the overall cost remains within an acceptable range.

7. Conclusion

This paper proposes FedCED, an innovative framework that overcomes the limitations of traditional KD-based methods in decentralized structures, significantly enhancing the performance of FL. This is achieved through information retention, relay-enhanced model collaboration, and historical information capture. In experiments, FedCED was benchmarked against existing methods (including FedAvg, FedProx, FedNKD, FedFTG, FedDCM, CaPriDe, and BRACE) under decentralized structures, demonstrating superior performance. Specifically, FedCED demonstrated improvements of up to 4.11% on the CIFAR-10, MNIST, and Fashion-MNIST datasets, showcasing its advancements. These results highlight FedCED's advantages in decentralized structures and its effectiveness in addressing data heterogeneity issues. In future work, we will focus on addressing the communication challenges faced by

FedCED, integrating it with model compression techniques (e.g. gradient quantization or pruning) to enhance training efficiency and further extend its application in this domain. In general, FedCED showsties in handling privacy-preserving decentralized data, offering an innovative approach for distributed training. This method not only opens new research avenues but also holds significant implications for applications in the FL field.

CRediT authorship contribution statement

Bin Liu: Funding acquisition. **Changle Li:** Writing – original draft, Data curation. **Qi Chu:** Visualization, Formal analysis, Data curation. **Banghao Zhai:** Validation. **Zeyu Ji:** Writing – review & editing. **Keqin Li:** Validation.

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 research is supported by the National Natural Science Foundation of China (No. 62376226), Shaanxi's Key Research and Development Program (No. 2024NC-ZDCYL-05-05), Xi'an Key Technology Research Projects for Key Agricultural Industry Chains (No. 2024JH-NYZD-0027), and the Yangling Demonstration Zone Science and Technology Plan Project (No. 2024NY-14).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.knosys.2026.116019.

Data availability

The authors do not have permission to share data.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, 2017, pp. 1273–1282.
[2] Y. Huang, L. Kong, Q. Li, B. Zhang, Decentralized federated learning via mutual knowledge distillation, in: IEEE International Conference on Multimedia and Expo, 2023, pp. 342–347.
[3] C. Che, X. Li, C. Chen, X. He, Z. Zheng, A decentralized federated learning framework via committee mechanism with convergence guarantee, IEEE Trans. Parallel Distrib. Syst. 33 (12) (2022) 4783–4800.
[4] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, Mach. Learn. Syst. 2 (2020) 429–450.
[5] J. Wang, Q. Liu, H. Liang, G. Joshi, H.V. Poor, Tackling the objective inconsistency problem in heterogeneous federated optimization, Adv. Neural Inf. Process. Syst. 33 (2020) 7611–7623.
[6] Q. Li, B. He, D. Song, Model-contrastive federated learning, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 10713–10722.
[7] Z. Zhu, J. Hong, J. Zhou, Data-free knowledge distillation for heterogeneous federated learning, in: International Conference on Machine Learning, 2021, pp. 12878–12889.
[8] L. Zhang, L. Shen, L. Ding, D. Tao, D. LingYu, Fine-tuning global model via data-free knowledge distillation for non-IID federated learning, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10174–10183.
[9] S. Zhu, Q. Qi, Z. Zhuang, J. Wang, H. Sun, J. Liao, Fednkd: A dependable federated learning using fine-tuned random noise and knowledge distillation, in: International Conference on Multimedia Retrieval, 2022, pp. 185–193.
[10] J. Zhang, C. Chen, W. Zhuang, L. Lyu, Target: Federated class-continual learning via exemplar-free distillation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 4782–4793.
[11] A. Zhmoginov, M. Sandler, N. Miller, G. Kristiansen, M. Vladymyrov, Decentralized learning with multi-headed distillation, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 8053–8063.
[12] B. Wang, Z. Tian, J. Ma, W. Zhang, W. She, W. Liu, A decentralized asynchronous federated learning framework for edge devices, Future Gener. Comput. Syst. 166 (2025) 107683.
[13] B. Li, W. Gao, J. Xie, H. Li, M. Gong, Robust decentralized federated learning for heterogeneous and non-ideal networks, Pattern Recognit. 162 (2025) 111362.
[14] H. Fu, F. Tian, G. Deng, L. Liang, X. Zhang, Reads: A personalized federated learning framework with fine-grained layer aggregation and decentralized clustering, IEEE Trans. Mob. Comput. 24 (8) (2025) 7709–7725.
[15] J. Shen, N. Cheng, X. Wang, F. Lyu, W. Xu, Z. Liu, K. Aldubaikhy, X. Shen, Ringsfl: An adaptive split federated learning towards taming client heterogeneity, IEEE Trans. Mob. Comput. 23 (5) (2023) 5462–5478.
[16] W. Li, Y. Peng, M. Du, F. Sun, X. Wang, L. Shen, Hypernetwork aggregation for decentralized personalized federated learning, in: Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, 2025, pp. 1440–1448.
[17] B. Soltani, V. Haghighi, Y. Zhou, Q.Z. Sheng, L. Yao, DFLStar: A decentralized federated learning framework with self-knowledge distillation and participant selection, in: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, 2024, pp. 2108–2117.
[18] P.M.S. Sánchez, E.T.M. Beltrán, M.F. Llamas, G. Bovet, G.M. Pérez, A.H. Celdrán, Profe: Communication-efficient decentralized federated learning via distillation and prototypes, in: IEEE International Conference on Communications, 2025, pp. 1596–1601.
[19] E. Jeong, M. Kountouris, Personalized decentralized federated learning with knowledge distillation, in: 2023 IEEE International Conference on Communications, 2023, pp. 1982–1987.
[20] X. Wang, G. Xiong, H. Cao, J. Li, Y. Liu, Decentralized federated learning with model caching on mobile agents, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 39, 2025, pp. 21296–21303.
[21] L. Wan, J. Ning, Y. Li, C. Li, K. Li, Intelligent fault diagnosis via ring-based decentralized federated transfer learning, Knowl.-Based Syst. 284 (2024) 111288.
[22] N. Tastan, K. Nandakumar, Capride learning: Confidential and private decentralized learning based on encryption-friendly distillation loss, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 8084–8092.
[23] M. Fang, Z. Liu, X. Zhao, J. Liu, Byzantine-Robust federated learning over ring-all-reduce distributed computing, in: Companion Proceedings of the ACM on Web Conference 2025, 2025, pp. 961–965.
[24] D.A.E. Acar, Y. Zhao, R.M. Navarro, M. Mattina, P.N. Whatmough, V. Saligrama, Federated learning based on dynamic regularization, 2021, arXiv preprint arXiv:2111.04263.
[25] L. Yuan, Z. Wang, L. Sun, P.S. Yu, C.G. Brinton, Decentralized federated learning: A survey and perspective, IEEE Internet Things J. 11 (21) (2024) 34617–34638.
[26] A. Krizhevsky, G. Hinton, et al., Learning Multiple Layers of Features from Tiny Images, Technical Report, 2009.
[27] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE (1998) 2278–2324.
[28] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017, arXiv preprint arXiv:1708.07747.
[29] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[30] G. Fang, J. Song, C. Shen, X. Wang, D. Chen, M. Song, Data-free adversarial distillation, 2019, arXiv preprint arXiv:1912.11006.
[31] M. Fang, Z. Zhang, Hairi, P. Khanduri, J. Liu, S. Lu, Y. Liu, N. Gong, Byzantine-Robust decentralized federated learning, in: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, 2024, pp. 2874–2888.
[32] L. Tian, H. Zhang, B. Liu, J. Zhang, N. Duan, A. Yuan, Y. Huo, VMF-SSD: A novel v-space based multi-scale feature fusion SSD for apple leaf disease detection, IEEE/ACM Trans. Comput. Biol. Bioinform. 20 (3) (2022) 2016–2028.
[33] L. Fei-Fei, R. Fergus, P. Perona, Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories, in: IEEE Conference on Computer Vision and Pattern Recognition, 2004, pp. 178–178.
[34] J. Stallkamp, M. Schlipsing, J. Salmen, C. Igel, The German traffic sign recognition benchmark: a multi-class classification competition, in: International Joint Conference on Neural Networks, 2011, pp. 1453–1460.