Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A Robust Federated Learning Method for Data Heterogeneity with Enhanced Momentum-guided Aggregation

Linhai Nie
Chengdu University
Jin Wang
wangjin@cdu.edu.cn

Chengdu University
naixuan Hu
Chengdu University

Research Article

Keywords: Federated Learning, Non-IID Data, Feature Partitioning, Gradient Momentum

Posted Date: November 28th, 2025

DOI: https://doi.org/10.21203/rs.3.rs-7978012/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A Robust Federated Learning Method for Data Heterogeneity with Enhanced Momentum-guided Aggregation

Linhai Nie[1], Jin Wang[1*], Naixuan Hu[1]

[1]College of Computer Science, Chengdu University, 2025 Chengluo Avenue, Chengdu, 610106, Sichuan, China .

*Corresponding author(s). E-mail(s): wangjin@cdu.edu.cn;
Contributing authors: 212024085404013@cdu.edu.cn; 212024085404007@cdu.edu.cn;

Abstract

Federated learning (FL) is an emerging distributed machine learning paradigm that enables multiple edge devices to collaboratively train a model for a specific task while preserving privacy. Yet, due to the non-independent and identically distributed (Non-IID) data dispersed on edge devices, FL suffers from slow convergence and low accuracy. This paper focuses on this problem, and proposes a robust method through enhanced data sharing. Specifically, this paper adopts feature distillation to obtain the performance sensitive features, which are used to generating proxy data for initializing public data before FL training. Meanwhile, We use a momentum-guided strategy for parameter aggregation. In order to evaluate the performance of the method, this paper also conducts many experiments. As demonstrated by the results, the method could outperform the state-of-the-art methods by 5.31% in terms of performance.

Keywords: Federated Learning, Non-IID Data, Feature Partitioning, Gradient Momentum

1 Introduction

FL is a distributed machine learning technique which could enable multiple devices to collaboratively train a global model while preserving data privacy. Federated Averaging (FedAvg) algorithm [1] is a typical FL framework, which allows multiple devices to train a machine learning model cooperatively while keeping the user data stored locally. FedAvg adopts the server-parameter model in which clients could exchange model parameters. Specifically, during one communication round of FedAvg, the clients use the local data to train a local model, and then the server aggregates the model parameters from the clients instead of the data. At the end of each round, the server sends the aggregated model back to all clients. Since FedAvg could enable multiple participants to collaboratively train a global model in a decentralized manner without leaking private data, it could satisfy the basic requirements of privacy protection and data security, and has attracted increasing attention in fields such as healthcare, medical analysis, and data security[2].

Although FL has shown great potential in real-world applications, heterogeneity problem arising from heterogeneous communication conditions, different computation capabilities, and Non-IID data [3] still challenges FL. Specifically, due to the significant difference of dataset in size, feature space, and label distribution of FL clients [4], the Non-IID data distribution would lead to inconsistencies in clients' local objective functions and optimization directions, thus creating a fundamental contradiction between minimizing local empirical loss and reducing global

empirical loss. It has been proven that data heterogeneity would introduce drift in clients‘ local updates, and further result in a deceleration in convergence speed [5, 6]. As shown by Zhao et al. [7], such parameter drift between a client‘s model and the centralized learning model originates from two aspects, i.e., residual parameter drift from the previous round and the gradient drift occurring in the current round. Thus, in a highly heterogeneous environment, FedAvg usually lacks a convergence guarantee, which only obtains compromised convergence speed and model performance.

In order to address the issue of Non-IID in FL, many research [7–9], have been conducted in terms of new learning objectives [8], aggregation strategies [9], and constructing shareable information across clients [7]. In particular, Ref. [7] show that limited data sharing between clients could significantly improve performance of FL. However, such limited data sharing may cause privacy issues to some extent [10–12]. Ref. [13] further proposed to counter such privacy concerns by adding noise to the shared data, yet it may incur performance degradation [14]. In order to effectively achieve a trade-off data privacy and model performance, Yang et al. [15] proposed a new framework FedFed through feature distillation, which partitioned local data into two parts, i.e., performance-robust and performance-sensitive features, and shared the performance-sensitive features with random noise. Actually, the above research only use first-order gradient descent to gradient update, yet ignoring the preceding iterations which might potentially accelerate convergence [16].

This paper aims to solve the Non-IID problem through momentum-guided aggregation and feature distillation. Specifically, we have designed an enhanced momentum-guided aggregation method that uses feature distillation to generate data to compensate for feature drift, and momentum-guided aggregation is applied during the server-side aggregation process to accelerate model convergence. Additionally, we have designed a delayed upload federated learning method to mitigate the impact of straggler nodes [17–19]. The main contribution of this paper is summarized as follows:

- We design an enhanced momentum-guided aggregation method that combines momentum-guided aggregation with feature distillation. Clients compensate for feature drift by generating data through feature distillation learning, and then use momentum-guided aggregation during the aggregation process to accelerate model convergence.
- We design a delayed update aggregation method to mitigate the influence of straggler in FL. Specifically, we set up scenarios where clients train locally for T_{local} rounds without uploading, and train for T_{update} rounds before uploading. We then compare three sets of T_{local} and T_{update} through experiments.
- We conducted comparative experiments. The results showed that, compared to the best-performing existing methods, the model's accuracy improved by 5.31%, and fewer rounds were required to reach the target accuracy.

The rest of the paper is organized as follows. In Sec. 2, we review existing approaches addressing heterogeneity issues in FL. Sec. 3 introduces the necessary background and concepts for FL. A detailed description of our proposed method is provided in Sec. 4. Experimental results and comparisons with other methods are presented in Sec. 5. Finally, we conclude the paper in Sec. 6.

2 Related work

FL was a distributed model training method that gained significant attention in recent years, especially in fields such as healthcare, medical analysis, and data security [2]. The FedAvg and FedSGD algorithms proposed by McMahan [1] were foundational and widely used FL algorithms. FedAvg updated the global model by averaging the client model parameters, while FedSGD, on the other hand, uploaded gradients and aggregated them on the server side. Although traditional FL frameworks such as FedAvg and FedSGD are effective for training distributed models, they only consider homogeneous models and do not address the need for user customization [20]. Traditional FL frameworks were also susceptible to the impact of system heterogeneity, where clients had different computation and communication capabilities, leading to the existence of straggler nodes and significant delays in runtime. In practical scenarios, the Non-IID nature of data and the presence of straggler nodes often made federated learning challenging to deploy.

To address the issue of unstable convergence and accuracy degradation caused by the Non-IID nature of the data, many studies have focused on optimizing the client models to reduce the discrepancy between

the client models and the global model. For example, FedProx [8] introduced a regularization term in the model update process to adjust the discrepancy between the client models and the server model; SCAFFOLD [5] used control variates to correct the local updates of the clients, preventing the local models from deviating from the global model, thus improving the convergence speed and stability of FL in Non-IID data environments; FedNova [21] used a normalized averaging method to eliminate target inconsistency while maintaining rapid convergence of errors; FedDyn [22] introduced dynamic regularization to alleviate the inconsistency between the local training objectives of clients and the global objective, thereby improving the convergence and model performance in heterogeneous data environments. However, most of these methods had limited impact on accelerating convergence rates and improving accuracy, making them less feasible for real-world FL deployments. Momentum-based optimization techniques were also explored for improving convergence stability and speed. Many methods introduced momentum SGD [23, 24] or adaptive gradient descent [25, 26] into the server model updates, or adopted global momentum correction for local updates, such as FedCM [27] and FedADC [28]. Based on these methods, FedACG implemented local updates with global model regularization using forward gradients, significantly improving efficiency without adding client storage and communication burdens.

In addition to optimizing the client models to address Non-IID data issues, a more direct approach was to share a small portion of private data. Zhao et al. [7] proposed sharing 5% of the data between edge devices as shared data, which could improve accuracy by 30%. However, this direct data-sharing approach could lead to privacy leakage. Therefore, how to share useful information while protecting privacy became an urgent challenge. To solve this problem, Afzali et al. [29] proposed a FL framework based on Conditional Generative Adversarial Networks (CGANs). This method shared knowledge by transmitting synthetic data generated by CGANs instead of directly transmitting the original model parameters, effectively protecting privacy compared to traditional model parameter transmission methods. The use of generated data demonstrated strong privacy protection advantages, but a key challenge lay in how to generate data that ensured global generalization while reflecting the personalized needs of clients. Yang et al. [15] employed data distillation techniques to partition features into sensitive and robust components. Sensitive features were used to generate proxy data, and differential privacy techniques were employed to protect this proxy data. Although this approach achieved good results, it still followed the traditional FL aggregation method, client parameters were uploaded to the server in each round, and the server distributed model parameters to the clients based on weighted averaging. This led to significant communication overhead, and there was still room for accelerating convergence speed.

In a traditional FL setup, the server must wait until the last update is received from the clients before starting aggregation. To accelerate the training speed [35], Asynchronous Federated Learning (AFL) methods have been adopted. However, AFL still faces scalability issues. To address this, FedBuff, a buffered asynchronous aggregation method, was proposed, which aims to jointly optimize the benefits of synchronous and AFL. The server performs aggregation only after receiving a specified number of updates. However, this method is still affected by straggler nodes. Chai et al. [36] proposed TiFL, which divides clients into multiple groups. To mitigate the impact of stragglers, TiFL groups clients based on computational efficiency and training latency. Clients are selected only from the same group, thus reducing the waiting time for the server. However, grouping clients based on efficiency and training latency increases the computational burden for the clients.

Although significant progress has been made in the research on Non-IID data and straggler nodes, challenges such as slow convergence and communication burdens still remain. To address these issues, we propose an enhanced momentum-guided aggregation method, which applies feature distillation to generate data that compensates for feature drift, and uses momentum-guided aggregation during the server-side aggregation process to accelerate model convergence. Additionally, a delayed upload strategy is adopted in the aggregation process to mitigate the impact of straggler nodes. This method not only improves model performance but also enhances privacy protection and accelerates convergence speed.

3 Preliminaries

3.1 Fundamentals of Federated Learning

Unlike traditional machine learning, FL does not require sharing raw data and allows clients to use standard algorithms to explore local patterns. It enables

clients to collaboratively train a global model without disclosing sensitive local data. The FL process typically involves four key steps.

(1) Local Training

Each client i uses its local data D_i to train local model parameters θ_i^t, the weight update for the local model is Eq. 1.

$$\theta_i^{t+1} = \arg\min_{\theta_i^t} \mathcal{L}_i(\theta_i^t; D_i) \tag{1}$$

where $\mathcal{L}_i(\theta_i; D_i)$ denotes the loss function at client i, and θ_i^{t+1} is the updated local model parameter.

(2) Model Update

Each client computes its local update as the difference between the new and previous model parameters, and the update is Eq. 2.

$$\Delta\theta_i^{t+1} = \theta_i^{t+1} - \theta_i^t. \tag{2}$$

This update is sent to the central server.

(3) Aggregation Update

The server aggregates local updates from all clients to compute the global update Δ^{t+1}. The updated global model can be written as Eq. 3.

$$\theta^{t+1} = \theta^t + \Delta^{t+1}. \tag{3}$$

(4) Global Model Distribution

After updating the global model parameter θ^{t+1}, the server sends it to all clients. Each client then continues local training or inference based on the received global model. The global model distribution can be written as Eq. 4.

$$\theta_i^{t+1} \leftarrow \theta^{t+1}. \tag{4}$$

In FL, FedAvg is a widely used algorithm that updates the global model by averaging parameter changes from all clients. Each client first uploads its local update $\Delta\theta_i^{t+1}$ to the server. The server then computes the global update by aggregating all received updates, and the global model update can be written as Eq. 5.

$$\Delta^{t+1} = \frac{1}{k}\sum_{i=1}^{k} \Delta\theta_i^{t+1}. \tag{5}$$

Where K represents the total number of clients in the FL framework.

4 Model Design

The entire process is divided into the training stage of the Variational Autoencoder(VAE) and the training stage of the deep model. In the VAE training stage, first, the client performs feature distillation to extract sensitive features, and the gradient transmission between the client and the server follows a delayed strategy. After the training is completed, the client uses the sensitive features to generate noise-perturbed proxy data, which is then uploaded to the server. In the deep model training stage, the client begins training the deep model using a mixture of local data and proxy data, with the gradient transmission between the client and the server also following a delayed strategy. On the server side, a momentum-guided aggregation method is used for model aggregation. As shown in Fig. 1.

In this section, we first explain how to partition sensitive and robust features, and how to add noise to the sensitive features in Sec. 4.1. In Sec. 4.2, we discuss techniques to accelerate model convergence by predicting the global lookahead gradient using gradient momentum. Finally, in Sec. 4.3, We propose our federated delayed aggregation strategy.

4.1 Feature Distillation

We introduce an information-theoretic approach [30], which extracts sensitive information from the learned representations to maintain the generalization ability of the global model while reducing redundancy in the shared data.

Specifically, x_i is a data sample in the batch, and y_i is the corresponding label of the data. we aim to extract a feature representation s_i, which is derived from the input x_i and contains the minimal sufficient information required to predict the label y_i. We express this theoretical process can be written as Eq. 6.

$$\min_{s_i} I(x_i; y_i|s_i), \quad \text{subject to} \quad I(x_i; s_i) \leq B. \tag{6}$$

Where s_i represents the feature representation extracted from the input x_i, $I(\cdot)$ denotes mutual information, which is used to quantify the information transfer between the input and the label, and B is a constant. The purpose of this formula is to allow the learned feature representation s_i to extract information relevant to the prediction task while filtering out irrelevant information. As shown in [30], the goal is, given the learned representation s_i, to minimize the mutual information between the input x_i and the label y_i, while ensuring that s_i retains only limited information about x_i.

In FL, the data distributions across different clients can vary significantly, and client data often exhibits

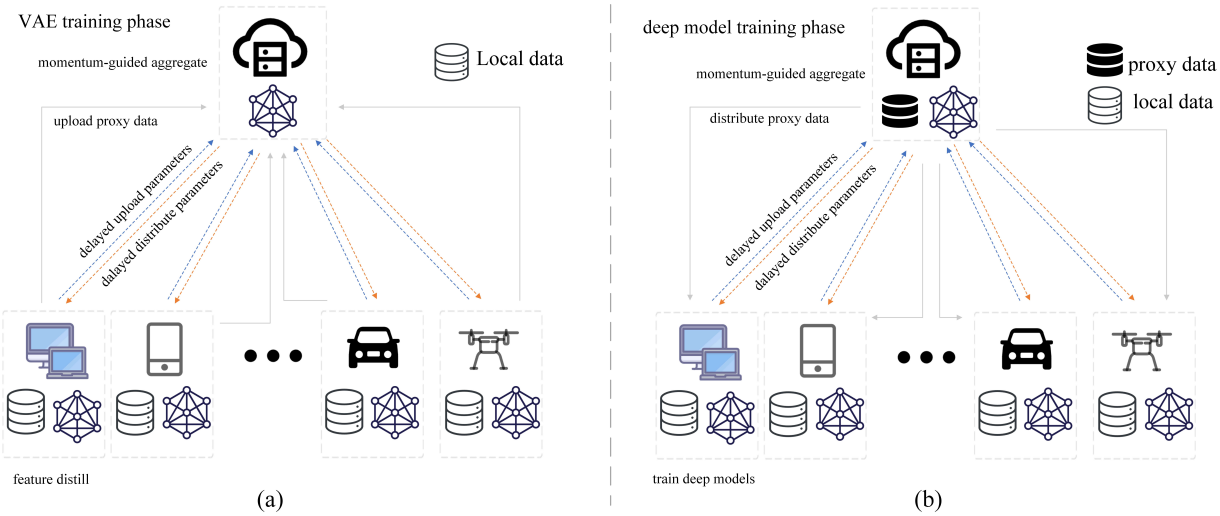


Fig. 1: The FL framework. (a) VAE training phase. (b) deep model training phase.

considerable heterogeneity. Our goal is to minimize the redundancy between the input data and its learned representation. To better manage the correlation and privacy of features, we divide the feature space into sensitive features and robust features. The sensitive features ,represented as s_i in Eq. 6, retain the unique characteristics of the data, which can be used to enhance the generalization ability of the global model. In contrast, the robust features capture the shared characteristics across clients. We only retain the sensitive features that contain the client's individuality, as this information is sufficient for accurate label prediction and improving overall model performance.

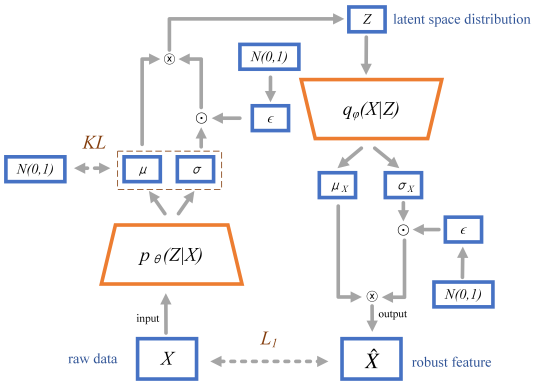


Fig. 2: The process of feature distillation.

To achieve this, we train a VAE on the client side and then aggregate to generate the global VAE on the server side. Since sensitive features reflect the client-specific characteristics, and each client has different personalized features, it is challenging for the global VAE to capture the sensitive features from each client. Therefore, we propose to train the VAE to learn robust features. Specifically, a batch contains N data samples, and the VAE samples the latent representation Z from the input data batch $X = \{x_1, x_2, \ldots, x_N\}$, and reconstructs the data $\hat{X} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N\}$ from Z, where $\hat{X}$ can be considered as the robust features. By calculating the difference between the original data X and the robust features $\hat{X}$, we can capture the features in the input data that are difficult to capture or easily influenced by perturbations, which we define as the sensitive features S. The sensitive feature S is defined by the Eq. 7.

$$S = \{s_1, s_2, \ldots, s_N\} = \{x_1 - \hat{x}_1, x_2 - \hat{x}_2, \ldots, x_N - \hat{x}_N\}. \tag{7}$$

The process of feature distillation is shown in Fig. 2. The VAE maps the input data batch X to the latent space Z through the encoder, and the decoder reconstructs the latent representation Z into robust features $\hat{X}$. The reconstruction loss is calculated as the mean squared error (MSE) between the input data batch X and the generated robust features $\hat{X}$. This loss ensures that the robust features $\hat{X}$ can effectively reconstruct the input data, thereby optimizing the globally shared features. The reconstruction loss can be expressed as Eq. 8.

$$L_1 = \sum_{i=1}^{N} |x_i - \hat{x}_i|_2^2. \tag{8}$$

KL divergence measures the difference between two probability distributions. In VAE, our goal is to compare the distribution of the latent variables with the standard normal distribution $N(0, 1)$. The KL divergence loss is computed as Eq. 9.

$$KL(Q(Z) \parallel N(0,1)) = -\frac{1}{2} \sum_{i=1}^{N} \left(1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2\right). \tag{9}$$

Where μ_i is the mean of the latent variables, σ_i^2 is the variance of the latent variables, and $Q(Z)$ is the latent space distribution predicted by the VAE model. The purpose of the KL divergence is to regularize the latent space so that it approximates the standard normal distribution, thus improving the model's generalization ability.

After the VAE outputs the robust features $\hat{X}$, the difference between X and $\hat{X}$ is computed to obtain the sensitive features S, and noise is added to S to generate synthetic data, as shown in Fig. 3. These proxy data are used to train the classifier. The cross-entropy loss measures the difference between the output generated by the noisy sensitive features and the true labels. The cross-entropy loss is defined as Eq. 10.

$$L_{\text{CE}} = -\sum_{i=1}^{N} y_i \log(p_i). \tag{10}$$

Where y_i is the probability distribution of the true labels, and p_i is the probability distribution predicted by the latent representation Z.

The total loss function is the weighted sum of the above losses and is defined as Eq. 11.

$$L_{\text{total}} = L_1 + \lambda_1 L_{\text{CE}} + \lambda_2 L_{\text{KL}}. \tag{11}$$

Where λ_1 and λ_2 are hyperparameters used to balance the contribution of each loss term to the total loss.

We use the FedAvg aggregation strategy to train the VAE. To reduce data redundancy and enhance model stability, we only upload the model components that are most relevant to label prediction, without disclosing excessive private information from the local data. To reduce the risk of privacy leakage during the process of uploading client model updates, we introduce differential privacy into the process. Specifically, we add Gaussian noise to the sensitive feature data, which can be expressed by the Eq. 12.

$$X_r = X + n. \tag{12}$$

Where X represents the sensitive feature data, n denotes the added Gaussian noise, and X_r is the proxy data uploaded to the server. As shown in Fig. 3, the image on the right is the noisy version of the image on the left. To evaluate the effectiveness of the added noise, we conducted a model inversion attack to attempt data reconstruction, as shown in Fig. 4. The figure shows the results of applying the model inversion attack on the noisy image. It can be observed that the attacker struggles to recover the original data. Each client generates proxy data based on the extracted sensitive features and uploads it to the server. The server only has access to the proxy data, while the original data remains securely stored on the client side, preparing for subsequent federated training of deep models.



(a) (b)

Fig. 3: (a) sensitive feature data. (b) proxy data.

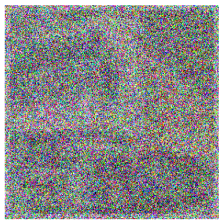


Fig. 4: The result of applying a model inversion attack on the image in (b) of Fig. 3.

4.2 Momentum-Guided Aggregation

After feature distillation, each client's VAE generates proxy data using the sensitive features, and this proxy data is shared across all clients. Then, the clients use both local data and the proxy data to train the deep learning model. To make the global model converge faster and more stably, we propose a method called *momentum-guided aggregation* when aggregating the global model. The core idea is to use the "global gradient information" from the global model to correct the initial state of the client models, thereby reducing their initial discrepancy, improving model stability, and accelerating convergence.

However, in FL, because the client models are decentralized and trained independently, we cannot directly access the exact global gradients. To address this issue, we introduce a momentum vector v, whose update rule is given by the Eq. 13.

$$v^{t+1} := \lambda v^t + \Delta^{t+1}. \tag{13}$$

Where Δ^{t+1} is defined by Eq. 5, and λ is a constant. This momentum vector accumulates the global model updates and serves as an effective approximation of the global gradient, capturing the overall trend of model evolution and effectively guiding the direction of future updates. Specifically, since our federated optimization framework does not perform global aggregation after each communication round, we aggregate and update the global model only after a certain number of rounds. The update rule is given by Eq. 14.

$$\theta^{t+1} = \theta^t + v^{t+1}. \tag{14}$$

By broadcasting the global model to the participating clients, these clients use it as the initialization for their local training, thus accelerating the convergence of the model. This method also maintains communication efficiency while enhancing the consistency of client training. In summary, the proposed method significantly accelerates the convergence of the global model by guiding the initial update and subsequent adjustments.

4.3 Delayed Transmission Strategy

In this study, we propose an enhanced version of the FedAvg algorithm in terms of aggregation to address the issue of straggler nodes caused by multiple clients.

Algorithm 1 Phase-based Aggregation Mechanism

Require: θ_0, C, T_{upload}, T_{local}, proxy datasets D^s, local datasets D^{C_i}
Ensure: Global model θ

for $k = 1, 2, \ldots, C$ **do**
  if first round **then**
    Sample subset of clients $S_t \subseteq \{C_1, \ldots, C_N\}$
    for each client $C_i \in S_t$ **in parallel do**
      $(w_k^i, \theta_k^i) \leftarrow \text{CTrain}(C_i, D^{C_i}, w_{k-1}, \theta_{k-1})$
      Send (w_k^i, Δ_k^i) to server
    end for
    $\theta_k \leftarrow \text{AGG}(w_k^i, \theta_k^i)$
    Server sends global model θ_k to clients
  else
    if $k \in T_{\text{upload}}$ **then**
      $(w'_k, \theta'_k) \leftarrow \text{STrain}(D^s, w_{k-1}, \theta_{k-1})$
      Server sends global model θ'_k to clients
      Sample subset of clients $S_t \subseteq \{C_1, \ldots, C_N\}$
      for each client $C_i \in S_t$ **in parallel do**
        $(w_k^i, \theta_k^i) \leftarrow \text{CTrain}(C_i, D^{C_i}, w'_k, \theta'_k)$
        Send (w_k^i, Δ_k^i) to server
      end for
      $\theta_k \leftarrow \text{AGG}(w_k^i, \theta_k^i)$
      Server sends global model θ_k to clients
    else if $k \in T_{\text{local}}$ **then**
      Sample subset of clients $S_t \subseteq \{C_1, \ldots, C_N\}$
      for each client $C_i \in S_t$ **in parallel do**
        $(w_k^i, \theta_k^i) \leftarrow \text{CTrain}(C_i, D^{C_i}, w_{k-1}, \theta_{k-1})$
        if next round is T_{upload} **then**
          Send (w_k^i, Δ_k^i) to server
        end if
      end for
      if next round is T_{upload} **then**
        $\theta_k \leftarrow \text{AGG}(w_k^i, \theta_k^i)$
      end if
    end if
  end if
end for

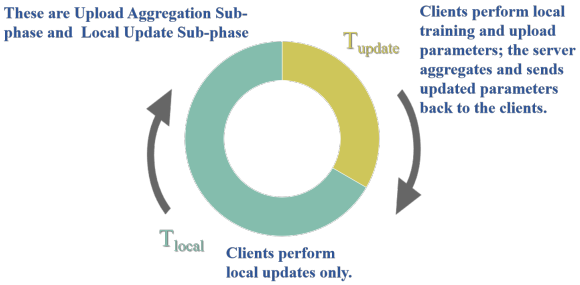


Fig. 5: Schematic of the aggregation strategy.

Table 1: Accuracy under different degrees of data heterogeneity, number of clients and local training epochs on CIFAR-10.

Method	**Acc**	**Round**	**Acc**	**Round**	**Acc**	**Round**	**Acc**	**Round**
	$\alpha = 0.1, N = 10$ $C_{Client} = 1$ $targetACC = 79\%$		$\alpha = 0.1, N = 10$ $C_{Client} = 5$ $targetACC = 85\%$		$\alpha = 0.05, N = 10$ $C_{Client} = 1$ $targetACC = 69\%$		$\alpha = 0.1, N = 100$ $C_{Client} = 1$ $targetACC = 49\%$	
FedAvg[1]	79.34	284	83.79	261	69.36	405	49.72	967
FedProx[8]	83.06	192	82.32	None	78.98	203	50.01	831
SCAFFOLD[5]	83.67	288	85.31	66	37.87	None	52.76	627
Fednova[21]	80.95	349	86.21	120	65.08	None	45.97	None
Fedfed(with FedAvg)[15]	90.29	52	88.89	33	85.10	38	80.92	233
Ours	**91.35**	**49**	**92.02**	**24**	**90.41**	**28**	**81.54**	**220**

Table 2: Accuracy under different degrees of data heterogeneity, number of clients and local training epochs on CIFAR-100.

Method	**Acc**	**Round**	**Acc**	**Round**	**Acc**	**Round**	**Acc**	**Round**
	$\alpha = 0.1, N = 10$ $C_{Client} = 1$ $targetACC = 67\%$		$\alpha = 0.1, N = 10$ $C_{Client} = 5$ $targetACC = 69\%$		$\alpha = 0.05, N = 10$ $C_{Client} = 1$ $targetACC = 61\%$		$\alpha = 0.1, N = 100$ $C_{Client} = 1$ $targetACC = 48\%$	
FedAvg[1]	67.64	286	68.22	267	60.24	424	47.51	988
FedProx[8]	63.54	201	61.54	None	61.37	474	47.62	844
SCAFFOLD[5]	66.59	298	57.74	None	68.46	199	50.04	654
Fednova[21]	66.97	475	58.62	None	70.98	274	47.03	None
Fedfed(with FedAvg)[15]	64.92	None	66.54	None	64.16	242	60.12	432
Ours	**69.58**	**140**	**69.45**	**297**	**68.47**	**63**	**65.39**	**238**

To solve this issue, we adopt a delayed update aggregation method and divide the training process into distinct phases. The process is as follows.

(1) Initialization Phase

In the initial round of FL, each client performs C_{client} local training steps and uploads its model update to the server before the round concludes. The server then applies a global momentum-based aggregation strategy to integrate these updates and accelerate convergence.

(2) Periodic Aggregation Phase

Beginning with the second round, training proceeds cyclically, alternating between global aggregation and purely local updates. As shown in Fig. 5, each cycle consists of the Upload Aggregation Sub-phase and the Local Update Sub-phase.

At the start of each round in upload aggregation sub-phase, the server first performs C_{server} iterations of global model updates based on the current parameters to enhance the model's generalization ability. The updated global model is then sent to the selected clients.

Each client receives the global model, performs C_{client} local training steps, and uploads its parameter updates to the server. The server aggregates all received updates to produce the global model for the next round.

In the subsequent T_{local} rounds, clients continue performing C_{client} local training steps without uploading any updates to the server. These local updates are retained on each client for continued training, significantly reducing communication costs.

(3) Cyclic Execution

The above cycle of $T_{\text{upload}} + T_{\text{local}}$ rounds is repeated iteratively until the global model converges.

This phase-based aggregation strategy effectively reduces communication frequency between clients

Table 3: Accuracy under different degrees of data heterogeneity, number of clients and local training epochs on SVHN.

Method	Acc	Round	Acc	Round	Acc	Round	Acc	Round
	α = 0.1,N =10 C_{Client} = 1 $targetACC$ = 88%		α = 0.1,N =10 C_{Client} = 5 $targetACC$ = 87%		α = 0.05,N =10 C_{Client} = 1 $targetACC$ = 82%		α = 0.1,N =100 C_{Client} = 1 $targetACC$ = 89%	
FedAvg[1]	87.62	296	86.67	138	81.76	376	88.21	684
FedProx[8]	85.32	None	80.17	None	77.02	None	87.74	664
SCAFFOLD[5]	81.21	None	78.86	None	73.68	None	87.69	694
Fednova[21]	87.74	296	88.06	194	81.35	791	81.05	None
Fedfed(with FedAvg)[15]	92.16	73	88.49	78	88.21	179	92.61	745
Ours	**93.05**	**53**	**90.39**	**41**	**88.23**	**154**	**93.46**	**148**

and the server while preserving local updates during each aggregation cycle. By alternating between communication-based aggregation and purely local updates, the proposed method achieves significant performance improvements while maintaining high efficiency. This design makes it more suitable for FL tasks in real-world scenarios characterized by low client participation and high data heterogeneity. The detailed procedure of the algorithm is outlined in Algorithm 1.

Table 4: Accuracy under different momentum coefficients on CIFAR-10.

Method	Acc	Round
	α = 0.1, N = 10 C_{Client} = 1 $targetACC$ = 88%	
Fedfed (with FedAvg)	90.29	52
Ours (λ = 0.2)	91.24	**49**
Ours (λ = 0.5)	**91.61**	137
Ours (λ = 0.8)	90.79	198

5 Results

To evaluate the effectiveness of the proposed method, we conducted a series of experiments on the CIFAR-10 [31], CIFAR-100 [31], and SVHN [32] datasets using VAE model[33] and ResNet-18 model[34]. Unless otherwise specified, all methods use the same default hyperparameters. Details of the experimental setup and results are presented in the following sections.

5.1 Evaluation Setup

All experiments are performed on a machine equipped with a single NVIDIA A100 GPU, an AMD Ryzen 7 9700X CPU, and 128 GB of RAM. The operating system is Windows 11, and the experiments are developed using PyCharm Professional Edition 2024.3.3 with PyTorch 2.4+ (GPU version).

Datasets Three datasets are used to evaluate the proposed method.

(1) CIFAR-10

The CIFAR-10 dataset contains 60,000 color images of size 32×32, divided into 10 distinct classes. It includes 50,000 training images and 10,000 test images. The dataset is widely used in image classification, covering diverse categories such as airplanes, automobiles, birds, and cats.

(2) CIFAR-100

The CIFAR-100 dataset resembles CIFAR-10 but includes 100 classes, each with 600 images. Like CIFAR-10, all images are 32×32 pixels. It provides 50,000 training images and 10,000 test images. The larger number of classes makes CIFAR-100 a more demanding benchmark for classification tasks.

(3) SVHN

The SVHN dataset contains over 600,000 labeled digits cropped from street view images. It is divided into three parts: a training set, a test set, and an additional set. The task is to classify digits from 0 to 9 in natural scenes, making SVHN a challenging benchmark due to its diverse and noisy real-world sources.

Non-IID Data Generation We generate Non-IID data following Hsu et al.'s method, sampling from a

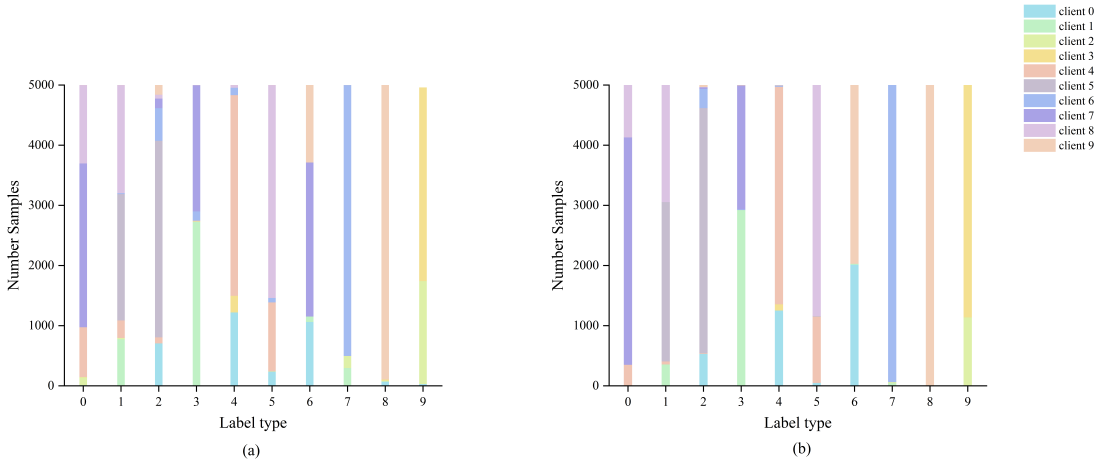


Fig. 6: the data distribution of the CIFAR-10 dataset under α values of 0.1 and 0.05. (a) $\alpha = 0.1$, (b) $\alpha = 0.05$

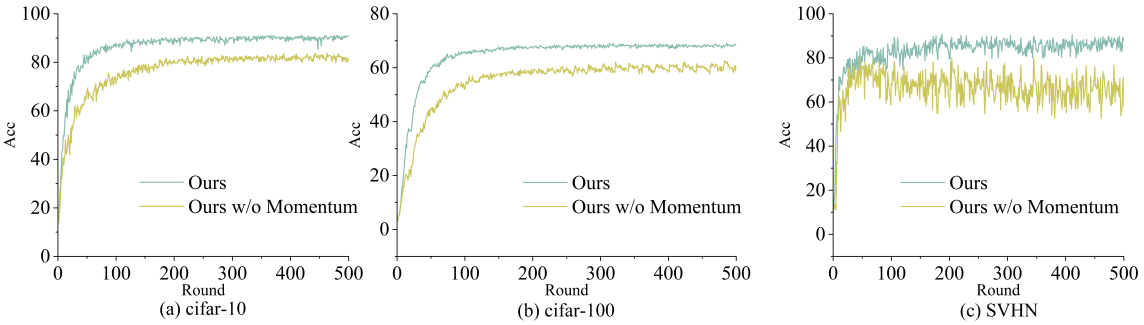


Fig. 7: Accuracy variations with and without momentum guidance on CIFAR-10.

Dirichlet distribution Dir(α), where $\alpha \in (0, \infty)$ controls the degree of heterogeneity. A larger α yields more balanced data across clients, while a smaller α produces more skewed distributions. In our experiments, we set $\alpha = 0.05$ and $\alpha = 0.1$ to test different levels of Non-IID settings. Fig. 6 illustrates the data distributions on CIFAR-10 for $\alpha = 0.1$ and $\alpha = 0.5$.

Experiments Setup Each client uses both a VAE and a ResNet-18 model, trained with the Adam optimizer. We set the learning rate to 0.001 for the VAE and 0.01 for the ResNet-18. All experiments are implemented in Python using the PyTorch framework.

Comparison Plan In this study, we compare our method with five FL approaches: FedAvg, FedProx, SCAFFOLD, FedNova, and FedFed (with FedAvg). **FedAvg** serves as the baseline method, which averages model updates across clients. We use it as a benchmark to evaluate how well our method handles heterogeneous data. **FedProx** extends **FedAvg** by adding a proximal term to better address client heterogeneity. **SCAFFOLD** introduces control variates to stabilize client updates, mitigating the impact of Non-IID data. **FedNova** normalizes client updates to improve convergence in the presence of data heterogeneity. Finally, **FedFed (with FedAvg)** focuses on extracting sensitive features to generate proxy data and applies a momentum-free federated averaging strategy. These comparisons enable us to highlight the strengths and weaknesses of our proposed method.

We use classification accuracy as a metric to reflect the performance of the framework on the test dataset. Accuracy is defined as Eq. 15.

$$\text{Acc} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(\hat{y}_i = y_i). \tag{15}$$

Table 5: Accuracy variations of FedFed and our methods under different T_{update} and T_{local} settings on CIFAR-10.

Method	Acc	Round	Acc	Round	Acc	Round	Acc	Round
	$\alpha = 0.1, N = 10$ $C_{Client} = 1$ $targetACC = 79\%$		$\alpha = 0.1, N = 10$ $C_{Client} = 5$ $targetACC = 85\%$		$\alpha = 0.05, N = 10$ $C_{Client} = 1$ $targetACC = 69\%$		$\alpha = 0.1, N = 100$ $C_{Client} = 1$ $targetACC = 49\%$	
Fedfed(with FedAvg)	90.29	52	88.89	33	85.10	38	80.92	233
Ours $T_{update} = 3$ $T_{local} = 3$	91.21	**44**	**92.02**	**24**	90.40	32	77.03	242
Ours $T_{update} = 10$ $T_{local} = 10$	91.15	67	91.93	25	**90.41**	**28**	**81.54**	**220**
Ours $T_{update} = 10$ $T_{local} = 20$	**91.35**	49	91.89	33	90.23	38	75.43	333

Table 6: Accuracy variations of FedFed and our methods under different T_{update} and T_{local} settings on CIFAR-100.

Method	Acc	Round	Acc	Round	Acc	Round	Acc	Round
	$\alpha = 0.1, N = 10$ $C_{Client} = 1$ $targetACC = 67\%$		$\alpha = 0.1, N = 10$ $C_{Client} = 5$ $targetACC = 69\%$		$\alpha = 0.05, N = 10$ $C_{Client} = 1$ $targetACC = 61\%$		$\alpha = 0.1, N = 100$ $C_{Client} = 1$ $targetACC = 48\%$	
Fedfed(with FedAvg)	64.92	None	66.54	None	64.16	242	60.12	432
Ours $T_{update} = 3$ $T_{local} = 3$	69.20	178	68.35	None	67.93	86	62.03	381
Ours $T_{update} = 10$ $T_{local} = 10$	69.13	167	**69.45**	**297**	**68.47**	63	64.31	284
Ours $T_{update} = 10$ $T_{local} = 20$	**69.58**	**140**	68.92	None	68.38	**53**	**65.39**	**238**

Where N is the size of the test set, $\mathbb{I}$ is the indicator function, which is 1 when $\hat{y}_i = y_i$ and 0 otherwise, $\hat{y}_i$ is the predicted label, and y_i is the true label.

5.2 Comparative Experiments

The experimental results on the CIFAR-10, CIFAR-100, and SVHN datasets are shown in Tab. 1, 2, 3, respectively. We observe that our method achieves higher accuracy, particularly when compared to FedFed (with FedAvg). Our method requires fewer communication rounds to reach the target accuracy, indicating that momentum-guided aggregation accelerates convergence. Specifically, on the CIFAR-10 dataset (Table 1) with $\alpha = 0.05$, $N = 10$, and $C_{\text{client}} = 1$, our method achieves significantly improved accuracy. On the CIFAR-100 dataset, using the same settings, our method accelerates convergence significantly. Our method shows limited improvement on the SVHN dataset, which could be due to the fact that the SVHN dataset mainly consists of digits from 0 to 9, with relatively simple features and a high amount of background noise, making feature extraction more challenging.

5.3 Hyperparameter Tuning Experiment

We conduct experiments with different values of T_{upload} and T_{local}, dividing them into three groups. The first group sets both T_{upload} and T_{local} to small values with $T_{\text{upload}} = T_{\text{local}}$. The second group uses larger values for both, maintaining the same equality. The third group also uses large values, but sets $T_{\text{upload}} \neq T_{\text{local}}$. We observed the changes in accuracy for each configuration. Tab. 5, Tab. 6, and Tab. 7 present the results on the CIFAR-10, CIFAR-100, and SVHN datasets, respectively.

We observe that our method outperforms existing methods under different combinations of T_{upload} and T_{local}, demonstrating higher accuracy and faster convergence in various scenarios. For all three datasets, when $\alpha = 0.05$ and $N = 100$, performance degrades when both T_{update} and T_{local} are set to 3, compared to the other two settings. This may be because, on more complex datasets, as heterogeneity increases or the number of clients grows, allowing clients to perform more local training rounds before uploading parameters enables them to learn richer representations, thereby enhancing overall performance. For the

Table 7: Accuracy variations of FedFed and our methods under different T_{update} and T_{local} settings on SVHN.

Method	Acc	Round	Acc	Round	Acc	Round	Acc	Round
	$\alpha = 0.1, N = 10$ $C_{Client} = 1$ $targetACC = 88\%$		$\alpha = 0.1, N = 10$ $C_{Client} = 5$ $targetACC = 87\%$		$\alpha = 0.05, N = 10$ $C_{Client} = 1$ $targetACC = 82\%$		$\alpha = 0.1, N = 100$ $C_{Client} = 1$ $targetACC = 89\%$	
Fedfed(with FedAvg)	92.16	73	88.49	78	88.21	179	92.61	745
Ours $T_{update} = 3$ $T_{local} = 3$	87.37	None	**90.39**	**41**	86.39	153	92.63	215
Ours $T_{update} = 10$ $T_{local} = 10$	**93.05**	**53**	88.44	51	86.93	**110**	**93.46**	148
Ours $T_{update} = 10$ $T_{local} = 20$	90.58	168	87.53	106	**88.23**	154	93.38	**128**

CIFAR-10 and CIFAR-100 datasets, when $N = 10$, $\alpha = 0.1$, and $C_{\text{client}} = 5$, the results across all three configurations remain largely consistent. However, on the SVHN dataset, when both T_{update} and T_{local} are set to 3, performance improves significantly. This could be because, with 5 local iterations per training round, larger values of T_{update} and T_{local} lead the model to miss optimal conditions. Furthermore, our method shows a greater improvement in accuracy and convergence speed on the CIFAR-100 dataset compared to CIFAR-10 and SVHN. This may be due to the fact that the CIFAR-100 dataset contains 100 classes, while CIFAR-10 and SVHN datasets only contain 10 classes. Our method exhibits more advantages when the number of classes is higher and the heterogeneity is stronger.

We evaluate the accuracy of our method by varying λ, using $\alpha = 0.1$, $N = 10$, and $C_{\text{client}} = 1$ as fixed parameters. Here, λ controls the degree of momentum integration in the global model. The experimental results appear in Tab. 4. As shown in Tab. 4, although the model with $\lambda = 0.5$ achieves the highest accuracy, slightly higner than that with $\lambda = 0.2$, the former requires 137 training rounds to reach the target accuracy, whereas the latter only requires 49 rounds. This suggests that although the model with $\lambda = 0.5$ achieves slightly higher accuracy, its slower convergence makes it less efficient in reaching the target accuracy than the model with $\lambda = 0.2$.

5.4 Ablation Experiment

We replace the momentum-guided aggregation method with standard federated averaging to evaluate its impact under the settings $\alpha = 0.1$, $N = 10$, and $C_{\text{client}} = 1$ across all three datasets.

As shown in Fig. 7, removing the momentum-guided aggregation leads to a marked drop in performance across all three datasets. The performance drop is most pronounced on the SVHN dataset, indicating that the momentum-guided method we adopted is effective. Our method even underperforms FedFed on this dataset. A possible reason is that our method uses a sporadic upload mechanism, whereas FedFed uploads parameters to the server for aggregation after each training round, resulting in better performance for FedFed.

6 Conclusion

In this study, we propose an enhanced momentum-guided aggregation method to mitigate the challenge of data heterogeneity in FL. The method extracts sensitive features from local data to generate proxy datasets, which is uploaded to the server, while the original data remains on the local clients. This design enables global knowledge sharing while maintaining data privacy. During global model aggregation, we implement a momentum-guided strategy to improve update stability. The momentum vector accumulates changes in the global model across previous rounds, aiming to enhance stability and accelerate convergence. In FL, clients are not required to communicate with the server in every round. Instead, a delayed transmission strategy is adopted, where clients synchronize with the server only at predefined intervals to reduce the impact of straggler nodes on the global model. We evaluate the proposed method on multiple datasets. The results demonstrate that it effectively mitigates slow convergence and low accuracy caused by Non-IID data. Future work will focus on deploying and validating the method in real-world scenarios. We aim to investigate its robustness and generalization under realistic

Non-IID data and dynamic client participation, further enhancing its practical value.

References

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, *Communication-Efficient Learning of Deep Networks from Decentralized Data*, in Proc. Int. Conf. on Artificial Intelligence and Statistics, 2016.

[2] M. Ye, X. Fang, B. Du, P. C. Yuen, and D. Tao, *Heterogeneous Federated Learning: State-of-the-art and Research Challenges*, ACM Computing Surveys, vol. 56, no. 3, pp. 79:1–79:44, Oct. 2023, doi: 10.1145/3625558.

[3] T. M. Hsu, H. Qi, and M. Brown, *Federated Visual Classification with Real-World Data Distribution*, in Proc. of the Computer Vision – ECCV 2020, Cham, Switzerland, Springer, 2020, pp. 76–92, doi: 10.1007/978-3-030-58607-2_5.

[4] F. Dong, H. Leung, and S. Drew, *Navigating High-Degree Heterogeneity: Federated Learning in Aerial and Space Networks*, in Proc. of the 2024 IEEE 10th World Forum on Internet of Things (WF-IoT), Ottawa, ON, Canada, IEEE, 2024, pp. 364–369, doi: 10.1109/WF-IoT62078.2024.10811460.

[5] S. P. Karimireddy, S. Kale, M. Mohri, and others, *SCAFFOLD: Stochastic Controlled Averaging for Federated Learning*, in Proc. of the 37th Int. Conf. on Machine Learning (ICML), 2020, pp. 5132–5143.

[6] L. Gao, H. Fu, L. Li, Y. Chen, M. Xu, and C. Z. Xu, *FedDC: Federated Learning with Non-IID Data via Local Drift Decoupling and Correction*, in Proc. of the 2022 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 10102–10111.

[7] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, *Federated Learning with Non-IID Data*, arXiv:1806.00582, 2018.

[8] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, *Federated Optimization in Heterogeneous Networks*, in Proc. of Machine Learning and Systems, 2020, pp. 429–450.

[9] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, *Bayesian Nonparametric Federated Learning of Neural Networks*, in Proc. of the 36th Int. Conf. on Machine Learning (ICML), Long Beach, CA, USA, 2019, pp. 7252–7261.

[10] M. Nasr, R. Shokri, and A. Houmansadr, *Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks Against Centralized and Federated Learning*, in Proc. of the 2019 IEEE Symp. on Security and Privacy (SP), San Francisco, CA, USA, IEEE, 2019, pp. 739–753, doi: 10.1109/SP.2019.00065.

[11] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, *A Survey on Security and Privacy of Federated Learning*, Future Generation Computer Systems, vol. 115, pp. 619–640, 2021, doi: 10.1016/j.future.2020.10.007.

[12] H. Yan, S. Li, Y. Wang, Y. Zhang, K. Sharif, H. Hu, and Y. Li, *Membership Inference Attacks Against Deep Learning Models via Logits Distribution*, IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 5, pp. 3799–3808, 2023, doi: 10.1109/TDSC.2022.3222880.

[13] M. Du, X. Yue, S. S. M. Chow, T. Wang, C. Huang, and H. Sun, *DP-Forward: Fine-tuning and Inference on Language Models with Differential Privacy in Forward Pass*, in Proc. of the 2023 ACM SIGSAC Conf. on Computer and Communications Security (CCS '23), Copenhagen, Denmark, ACM, 2023, pp. 2665–2679, doi: 10.1145/3576915.3616592.

[14] F. Tramer and D. Boneh, *Differentially Private Learning Needs Better Features (or Much More Data)*, in Proc. of the Fourteenth Int. Conf. on Learning Representations (ICLR), 2021.

[15] Z. Yang, Y. Zhang, Y. Zheng, X. Tian, H. Peng, T. Liu, and B. Han, *FedFed: Feature Distillation Against Data Heterogeneity in Federated Learning*, in Proc. of Advances in Neural Information Processing Systems (NeurIPS), New Orleans, LA, USA, 2023, vol. 36, eprint: arXiv:2310.05077.

[16] W. Liu, L. Chen, Y. Chen, and W. Zhang, *Accelerating Federated Learning via Momentum Gradient*

Descent, IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 8, pp. 1754–1766, 2020.

[17] Z. Chen, W. Yi, H. Shin, and A. Nallanathan, *Adaptive Model Pruning for Communication and Computation Efficient Wireless Federated Learning*, IEEE Transactions on Wireless Communications, vol. 23, no. 7, pp. 7582–7598, 2024, doi: 10.1109/TWC.2023.3342626.

[18] Y. Wang, N. Li, X. Wang, F. Zhong, and et al., *Coding-Based Performance Improvement of Distributed Machine Learning in Large-Scale Clusters*, Journal of Computer Research and Development, vol. 57, no. 3, pp. 542–561, 2020, doi: 10.7544/issn1000-1239.2020.20190286.

[19] G. Kim, J. Kim, and B. Han, *Communication-Efficient Federated Learning with Accelerated Client Gradient*, in Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, IEEE, 2024, pp. 12385–12394, doi: 10.1109/CVPR52733.2024.01177.

[20] C. Sun, T. Jiang, and D. Pompili, *Heterogeneous Federated Learning via Generative Model-Aided Knowledge Distillation in the Edge*, IEEE Internet of Things Journal, vol. 12, no. 5, pp. 5589–5599, 2025, doi: 10.1109/JIOT.2024.3488565.

[21] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, *Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization*, in Proc. of the 34th Int. Conf. on Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, Curran Associates, Inc., 2020, pp. 1–13.

[22] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, V. Saligrama, and others, *Federated Learning Based on Dynamic Regularization*, arXiv:2111.04263, 2021.

[23] T. H. Hsu, H. Qi, and M. Brown, *Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification*, arXiv preprint arXiv:1909.06335, 2019. Available: https://arxiv.org/abs/1909.06335.

[24] J. Wang, V. Tantia, N. Ballas, and M. Rabbat, *SlowMo: Improving Communication-Efficient Distributed SGD with Slow Momentum*, arXiv preprint arXiv:1910.00643, 2020. Available: https://arxiv.org/abs/1910.00643.

[25] D. Caldarola, B. Caputo, and M. Ciccone, *Improving Generalization in Federated Learning by Seeking Flat Minima*, in *Computer Vision – ECCV 2022: 17th European Conference*, Tel Aviv, Israel, Oct. 2022, pp. 654–672.

[26] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, *Adaptive Federated Optimization*, in *International Conference on Learning Representations*, 2021. Available: https://openreview.net/forum?id=LkFG3lB13U5.

[27] J. Xu, S. Wang, L. Wang, and A. C.-C. Yao, *FedCM: Federated Learning with Client-level Momentum*, arXiv preprint arXiv:2106.10874, 2021, [Online]. Available: https://arxiv.org/abs/2106.10874.

[28] E. Ozfatura, K. Ozfatura, and D. Gündüz, *FedADC: Accelerated Federated Learning with Drift Control*, in *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 467-472, doi: 10.1109/ISIT45174.2021.9517850.

[29] A. Afzali and P. Shamsinejadbabaki, *Private and Heterogeneous Personalized Hierarchical Federated Learning Using Conditional Generative Adversarial Networks*, Expert Systems with Applications, vol. 276, 127003, 2025, doi: 10.1016/j.eswa.2025.127003.

[30] R. Shwartz-Ziv and N. Tishby, *Opening the Black Box of Deep Neural Networks via Information*, in Information Flow in Deep Neural Networks, Springer, Cham, Switzerland, 2022, pp. 1–28.

[31] A. Krizhevsky, G. Hinton, and others, *Learning Multiple Layers of Features from Tiny Images*, Technical Report, University of Toronto, 2009.

[32] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, *Reading Digits in Natural Images with Unsupervised Feature Learning*, in Proc. of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning, Granada, Spain,

2011, pp. 5.

[33] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, *β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework*, in Proc. of the Int. Conf. on Learning Representations (ICLR), 2017.

[34] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, in Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[35] Z. Wang, Z. Zhang, Y. Tian, Q. Yang, H. Shan, W. Wang, and T. Q. S. Quek, *Asynchronous Federated Learning Over Wireless Communication Networks*, in *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6961–6978, 2022, doi: 10.1109/TWC.2022.3153495.

[36] Z. Chai, A. Ali, S. Zawad, S. Truex, A. Anwar, N. Baracaldo, Y. Zhou, H. Ludwig, F. Yan, and Y. Cheng, *TiFL: A Tier-based Federated Learning System*, in *HPDC '20: Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, 2020, pp. 125–136, doi: 10.1145/3369583.3392686, [Online]. Available: https://doi.org/10.1145/3369583.3392686.

Statements and Declarations

- Funding
  The authors declare that no funds, grants, or other support were received in preparation of this manuscript.
- Conflict of interest/Competing interests
  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.
- Author contribution
  All authors contributed to the conceptualization and design of the research. Material preparation, data collection, and analysis were performed by Linhai Nie, Jin Wang, and Naixuan Hu. The first draft of the manuscript was written by Linhai Nie, and all authors reviewed and provided feedback on previous versions of the manuscript. All authors read and approved the final manuscript.
- Data availability.
  The datasets used in this study, including CIFAR-10, CIFAR-100, and SVHN, are publicly available and can be accessed from the following links:
  CIFAR-10:https://www.cs.toronto.edu/~kriz/cifar.html
  CIFAR-100:https://www.cs.toronto.edu/~kriz/cifar.html
  SVHN: http://ufldl.stanford.edu/housenumbers/
  These datasets are freely available for research purposes.