Transactions on Artificial Intelligence
https://www.sciltp.com/journals/tai

Article

FedA^4: Federated Learning with Anti-Bias Aggregation and TrAjectory-Based Adaptation

Guanyi Zhao [1], Juntao Hu [1,2], Zhengjie Yang [1,3,*] and Dapeng Wu [1]

[1] Department of Computer Science, City University of Hong Kong, Hong Kong
[2] School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China
[3] Hong Kong Generative AI Research and Development Center, The Hong Kong University of Science and Technology, Hong Kong
* Correspondence: zhengjie.yang@sydney.edu.au

How To Cite: Zhao, G.; Hu, J.; Yang, Z.; et al. FedA^4: Federated Learning with Anti-Bias Aggregation and TrAjectory-Based Adaptation. *Transactions on Artificial Intelligence* **2026**, *2*(1), 26–38. https://doi.org/10.53941/tai.2026.100003

Received: 2 January 2026
Revised: 29 January 2026
Accepted: 2 February 2026
Published: 9 February 2026

Abstract: Non-Independent and Identically Distributed (Non-IID) data pose a fundamental challenge in Federated Learning (FL). It usually causes a severe client drift issue (various client model update directions) and thus, degrades the global model performance. Existing methods typically address this by assigning appropriate weights to client models or optimizing model update directions. However, these methods overlook client model update trends. They focus solely on the final client models to be aggregated at the server at each communication round, ignoring model optimization trajectories, which may contain richer information to aid model convergence. To address this issue, we propose FedA^4, a novel FL framework with **A**nti-bias **A**ggregation and tr**A**jectory-based **A**daptation, which leverages clients' optimization trajectories, rather than only their final model snapshots. For anti-bias aggregation, by observing a phenomenon termed model collapse, where biased clients tend to predict any input data as the dominant classes in their own datasets, we quantify the class dominance and analyze the level of client drift for each client. We evaluate a prediction entropy, namely concentration, so as to assign an optimal weight to each client at each training round. To further mitigate the negative effect of clients with high levels of client drift (biased clients), we then develop a gradient adaptation mechanism termed trajectory-based adaptation, which analyzes clients' trajectories to correct each client's contribution to the aggregated global model. Extensive experiments on CIFAR-10, CIFAR-100, STL-10, and Fashion-MNIST demonstrate that FedA^4 significantly outperforms state-of-the-art baselines, particularly in scenarios with extreme data heterogeneity (high level of Non-IID).

Keywords: federated Learning; non-IID Data; client drift; anti-bias aggregation; optimization trajectory

1. Introduction

Federated Learning (FL) has emerged as a novel distributed machine learning paradigm, enabling models to learn from vast amounts of data distributed across various clients (such as mobile phones, hospitals, or financial institutions) without the need for centralized data storage [1,2]. This privacy-preserving characteristic grants it significant potential in data-sensitive domains. In a typical FL setting, a central server coordinates training among numerous clients: the server distributes a global model to the clients, which then perform local training on their own data. These updates (e.g., model weights or gradients) are then sent back to the server, which aggregates them to optimize the global model in an iterative fashion [1].

However, FL faces a core challenge: client drift caused by data heterogeneity [3]. The local data distributions on different clients are typically not independent and identically distributed (Non-IID), which results in the local optimization objectives of clients diverging from the global objective. For instance, an orthopedic hospital mainly collects bone-related images while an ophthalmology hospital collects ocular images. During a classification task, these two hospitals, acting as two clients, try to minimize their local classification losses; as a result, their model parameters evolve towards disparate local optima, drifting away from the global optimum. We classify clients as biased if their local datasets fail to represent the complete dataset, irrespective of data volume or label distribution. Consequently, models trained on such datasets inevitably exhibit bias toward specific local distributions, and this phenomenon is referred to as client drift [4]. Therefore, designing an aggregation algorithm to mitigate client drift under Non-IID conditions is a central research problem in the FL domain. Recent studies have explored three different technical paths to address client drift caused by Non-IID conditions.

Weighted Model Aggregation. This kind of strategy aims to mitigate the impact of biased clients by adaptively optimizing aggregation weights. While the seminal FedAvg [1] relies on data volume, subsequent studies incorporate performance-driven metrics, such as local loss [5], model similarity [6–8], or Euclidean distances [9]. However, mere weight adjustment fails to yield a globally representative model [10–12], leaving the aggregated model susceptible to recurrent drift in subsequent rounds.

Client Parameter Constraint. To address the limitations of weighted aggregation, researchers have focused on suppressing client drift by constraining local parameters to remain close to the global model. FedProx [13] exemplifies this approach by introducing a proximal term to the local loss function, thereby penalizing deviations between local and global parameters. SCAFFOLD [14] employs control variates to estimate and correct client drift during local training, while [15,16] utilizes a dynamic regularizer to align local and global objectives. Although these methods effectively mitigate the magnitude of client drift, the aggregated global model may still suffer from bias, particularly favoring clients with larger dataset volume or larger label counts. Furthermore, the inherent "pull-back" mechanism may overly restrict local learning; this can trap the global model in a local optimum and hinder convergence to a superior global optimum.

Generative Model-Assisted Global Training. To address client drift caused by Non-IID data, some studies attempt to generate a synthetic global dataset to adjust model parameters. For instance, FedGen [17] and FedLMG [18] train a server-side generator using client features to create "imaginary" data for global training. Since privacy rules prevent access to raw data, these methods rely on adversarial or style transfer methods. However, adversarial methods often suffer from training instability. The difficulty in balancing the game between the generator and discriminator often leads to mode collapse or non-convergence [19]. Meanwhile, style transfer methods are effective only for tasks compatible with specific image styles, which limits the general applicability of these framework.

To summarize existing methods, FL requires finding reasonable aggregation weights, constraining biased client parameters, and mimicking global training behavior (as if a single model was trained on a union of all clients' datasets). To address these needs, we propose FedA^4, a novel FL framework with **A**nti-bias **A**ggregation and tr**A**jectory-based **A**daptation mechanisms. As the cornerstone of FedA^4, we identify biased clients based on the "model collapse" phenomenon, where clients make overconfident errors causing predictions to concentrate on dominant labels due to local bias. The anti-bias aggregation module decreases the aggregation weights of these biased clients to derive the global model. Furthermore, prior research [20,21] indicates that changes in model parameters reflect the characteristics of the underlying data distribution. Leveraging this insight, we require clients to upload optimization trajectories consisting of the traces of changes for client models between local epochs. The trajectory-based adaptation module utilizes these trajectories to infer an update gradient that mimics the behavior of the global model trained on the complete dataset. Specifically, anti-bias aggregation establishes reasonable aggregation weights, while trajectory-based adaptation imitates training on the complete dataset and constrains the influence of. client drift on the aggregated model update. This approach combines the strengths of current methods while compensating for their shortcomings.

The workflow begins with the client-side operations, where clients upload local models and optimization trajectories. The core processing occurs in the server-side operations through three steps, with the first and second steps serving as prerequisites for the third step. First, the evaluation of anti-bias weights step utilizes a server-resident probe dataset to assess clients. We measure concentration via prediction entropy; low entropy indicates biased clients concentrating on dominant local classes, while high entropy signals unbiased models. Combined with accuracy to filter under-trained models, these metrics guide the aggregation strategy to assign higher weights to unbiased clients. We also set bias penalty weights to filter out under-trained models in this step. Second, the adaptation of client gradients step calculates the deviation between local and global gradients. Critically, we perform gradient ascent for biased clients to filter out their negative impact, while performing gradient descent for unbiased clients to reinforce shared knowledge. After gathering all information from the above two steps, the anti-bias aggregation and trajectory-based adaptation step aggregates client models and adapts the aggregated model gradient.

We conduct extensive experiments on CIFAR-10, CIFAR-100 [22], STL-10 [23], and Fashion-MNIST [24]. The results demonstrate that our method significantly outperforms existing baselines. This performance advantage is particularly pronounced across diverse settings, effectively accommodating heterogeneous client model architectures and varying numbers of clients.

The primary contributions of this paper are:

1. We present FedA^4, a novel FL framework with **A**nti-bias **A**ggregation and tr**A**jectory-based **A**daptation mechanisms. By introducing an entropy-based metric for model concentration, an accuracy-based bias penalty scheme, and a gradient correction deception mechanism, FedA^4 systematically refines the global aggregation and update. These mechanisms jointly ensure more precise optimization and avoid local optima.
2. We introduce the optimization trajectory as the information carrier from clients, demonstrating that the training process, not just the result, can be used by the server to more accurately diagnose and correct client drift.
3. FedA^4 demonstrates superior performance in every tested scenario, peaking on CIFAR-100 [22] with a substantial average improvement of 4.61% over baselines across diverse Dirichlet distribution configurations.

2. Related Work

Data heterogeneity (Non-IID) in FL is a key challenge that causes client drift and degrades global model performance [3]. To address this challenge, a substantial body of work has emerged, which can be broadly categorized into three primary streams: weighted model aggregation, model parameter constraints, and generative model-assisted global training.

2.1. Weighted Model Aggregation

Weighted aggregation methods attempt to mitigate the impact of biased clients by assigning higher weights to unbiased ones, aiming to steer global convergence. Seminal studies like FedAvg [1] weight clients based on data volume, assuming dataset size correlates with representativeness. However, in Non-IID environments, large local datasets may still exhibit severe bias. Subsequent strategies recognize the need to downweight poorly performing clients, positing that low performance often stems from label scarcity or data insufficiency. Approaches such as [25,26] dynamically adjust weights based on loss or validation accuracy, while FedAtt [6] and pFedMe [7] use attention mechanism to find better aggregation weights. Additionally, RFA [27] employs the geometric median to enhance robustness against outliers. Nevertheless, neither simple averaging nor sophisticated weighting fundamentally resolves client drift, as biased local models often lack consistent mathematical alignment and may yield conflicting updates. Consequently, aggregating these conflicting parameters fails to achieve global representativeness [10–12], leading to recurrent drift in subsequent rounds. In contrast, while our method employs weighted aggregation, it distinguishes itself by introducing an adaptation phase that actively updates global parameters using optimization trajectory insights.

2.2. Client Parameter Constraint

The second category focuses on mitigating client drift via local regularization or constraints. FedProx [13], a representative work, adds a proximal term to the local loss function to penalize parameter divergence from the global model. Alternatively, SCAFFOLD [14] employs control variates to correct update directions, while FedDyn [15] and MOON [28] utilize dynamic regularization and contrastive learning, respectively. However, these methods implicitly assume the global model serves as a correct reference anchor. Consequently, regardless of the specific technique, they remain susceptible to bias from clients with dominant data volume or label diversity, impelling the global model toward local optima. Furthermore, the inherent regularization restricts local exploration, potentially hindering convergence to the true global optimum. In contrast, FedA^4 addresses these limitations by incorporating the optimization trajectory to capture the specific learning intent driven by local distributions. By synthesizing these trajectories, we infer a holistic global optimization path to guide precise updates without constraining client-side learning.

2.3. Generative Model-Assisted Global Training

This category aims to approximate the global data distribution to assist global model training via data generation techniques. Methods like FedGAN [29], FedGen [17] and FedLMG [18] train generators (e.g., GANs or decoders) using uploaded gradients to synthesize pseudo-global data for server-side training. The server then uses this generated data to assist in training or calibrating the global model. These methods are conceptually significant as they seek a global optimization direction. While conceptually appealing for providing a global perspective, this approach faces significant challenges: due to privacy concerns, these methods cannot ask clients to upload any information about the local dataset; they can only use adversarial methods to generate data or generate data with different styles to train the global model. However, adversarial methods face difficulties in balancing the game between the generator and discriminator, which often leads to mode collapse or non-convergence [19]. At the same time, style transfer methods are effective only for tasks compatible with specific image styles, which limits the general applicability of these frameworks.

3. Methodology

The Non-Independent and Identically Distributed (Non-IID) nature of client data poses a core challenge in FL, often rendering standard weighted aggregation insufficient for global representativeness. Moreover, existing methods typically rely solely on the final model state for aggregation, discarding the significant information contained in intermediate local client parameter updates. To address these limitations, we propose FedA^4, a framework integrating two novel mechanisms, as illustrated in Figure 1.

First, by addressing the phenomenon where biased models exhibit high prediction concentration (model collapse), we develop anti-bias aggregation. This module leverages prediction entropy on a server-side probe dataset to identify and downweight biased clients. Second, to recapture the optimization dynamics lost by local training, we develop trajectory-based adaptation. By tracking the optimization trajectory across local epochs, this component approximates the global update gradient effectively, ensuring stability and generalization. The operational workflow of FedA^4 proceeds as follows: (3.1) Client-Side Operations: Clients upload their local model parameters and optimization trajectories to the central server. (3.2) Server-Side Operations: (3.2.1) Evaluation of Anti-bias Weights: The server analyzes the entropy of client prediction distributions to derive anti-bias weights for all clients, which comprise both aggregation weights and bias penalty weights; (3.2.2) Adaptation of Client Gradients: The server computes an aligned gradient for each client based on the optimization trajectory and the determined gradient direction (ascent or descent); (3.2.3) Anti-Bias Aggregation and Trajectory-Based Adaptation: The server first aggregates client models using the weights from (3.2.1) to obtain an initial global model (Phase I). Subsequently, it updates this model using the aligned gradients from (3.2.2) to integrate trajectory information (Phase II). This process repeats for multiple rounds until convergence. Step (3.1) executes in parallel across clients, while steps (3.2.1) to (3.2.3) occur sequentially on the server.

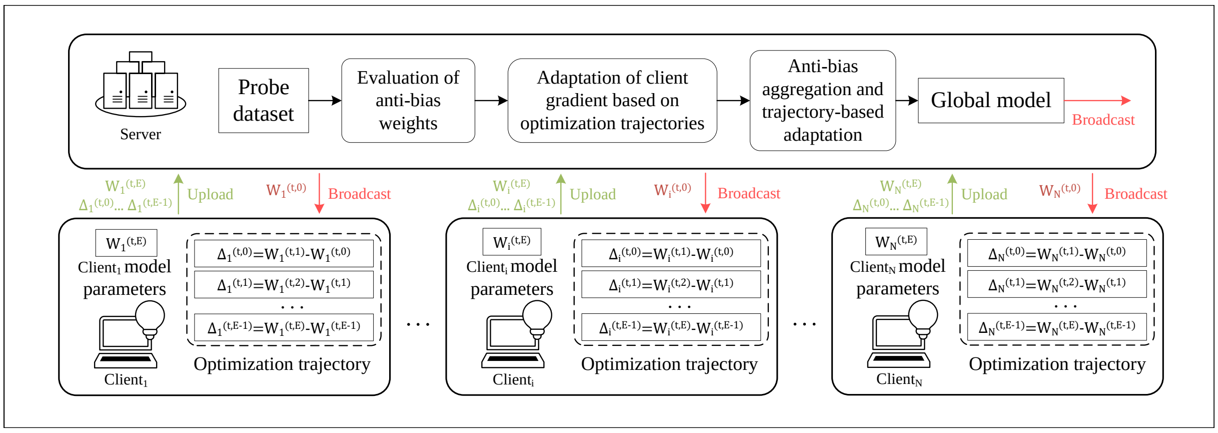

Server
Probe dataset
Evaluation of anti-bias weights
Adaptation of client gradient based on optimization trajectories
Anti-bias aggregation and trajectory-based adaptation
Global model
Broadcast
$W_1^{(t,E)}$ $\Delta_1^{(t,0)}\ldots\Delta_1^{(t,E-1)}$ Upload
$W_1^{(t,0)}$ Broadcast
$W_1^{(t,E)}$ Client$_1$ model parameters
$\Delta_1^{(t,0)}=W_1^{(t,1)}-W_1^{(t,0)}$
$\Delta_1^{(t,1)}=W_1^{(t,2)}-W_1^{(t,1)}$
$\Delta_1^{(t,E-1)}=W_1^{(t,E)}-W_1^{(t,E-1)}$
Optimization trajectory
Client$_1$
$W_i^{(t,E)}$ $\Delta_i^{(t,0)}\ldots\Delta_i^{(t,E-1)}$ Upload
$W_i^{(t,0)}$ Broadcast
$W_i^{(t,E)}$ Client$_i$ model parameters
$\Delta_i^{(t,0)}=W_i^{(t,1)}-W_i^{(t,0)}$
$\Delta_i^{(t,1)}=W_i^{(t,2)}-W_i^{(t,1)}$
$\Delta_i^{(t,E-1)}=W_i^{(t,E)}-W_i^{(t,E-1)}$
Optimization trajectory
Client$_i$
$W_N^{(t,E)}$ $\Delta_N^{(t,0)}\ldots\Delta_N^{(t,E-1)}$ Upload
$W_N^{(t,0)}$ Broadcast
$W_N^{(t,E)}$ Client$_N$ model parameters
$\Delta_N^{(t,0)}=W_N^{(t,1)}-W_N^{(t,0)}$
$\Delta_N^{(t,1)}=W_N^{(t,2)}-W_N^{(t,1)}$
$\Delta_N^{(t,E-1)}=W_N^{(t,E)}-W_N^{(t,E-1)}$
Optimization trajectory
Client$_N$

Figure 1. Overview of FedA^4 framework.

3.1. Client-Side Operations

In each global round $t \in [0, T-1]$, each client i initializes its local model $W_i^{t,0}$ using the global parameters W_{global}^t. Subsequently, the client performs local training on its private dataset $\mathcal{D}_i$ for E epochs to minimize the local objective $\mathcal{L}_i$ (Algorithm 1, Lines 2–6). Distinct from standard protocols, each client records the optimization trajectory P_i^t (Algorithm 1, Line 7) rather than solely retaining the final model state. We define the parameter update at local epoch e as $\Delta_i^{t,e} = W_i^{t,e+1} - W_i^{t,e}, \forall e \in [0, E-1]$. Consequently, the complete optimization trajectory is denoted as:

$$P_i^t = [\Delta_i^{t,0}, \Delta_i^{t,1}, ..., \Delta_i^{t,E-1}]. \tag{1}$$

This trajectory encodes the gradient dynamics induced by local data heterogeneity. Upon completing E epochs, client i uploads both the final parameters $W_i^{t,E}$ and the trajectory P_i^t to the server for global aggregation.

Algorithm 1 Client Update Procedure

```
Input: Client index $i$, global model $W^t_{\text{global}}$, local dataset $\mathcal{D}_i$, number of epochs $E$, learning rate $\gamma$
Output: Updated local model $W_i^{t,E}$ and optimization path $P_i^t$
 1: function ClientUpdate($i, W^t_{\text{global}}, \mathcal{D}_i, E$)
 2:     $W_i^{t,0} \leftarrow W^t_{\text{global}}$                          ▷ Initialize Local Model
 3:     $P_i^t \leftarrow [\,]$                                   ▷ Initialize Empty Optimization Path
 4:     for each local epoch $e = 0$ to $E-1$ do
 5:         $\Delta_i^{t,e} = -\gamma \nabla \mathcal{L}_i(W_i^{t,e}, \mathcal{D}_i)$           ▷ Calculate $\Delta_i^{t,e}$
 6:         $W_i^{t,e+1} \leftarrow W_i^{t,e} + \Delta_i^{t,e}$
 7:         Append $\Delta_i^{t,e}$ to $P_i^t$                        ▷ Record the Path
 8:     end for
 9:     return $W_i^{t,E}, P_i^t$                     ▷ Return Final State and Full Optimization Path
10: end function
```

3.2. Server-Side Operations

Upon collecting $W_i^{t,E}$ and P_i^t from all N clients in round t, the server initiates a global model aggregation and update framework that is divided into three distinct stages. In the first stage, the server evaluates client models to derive anti-bias weights by calculating the aggregation weights based on prediction concentration and the bias penalty weights based on the deviation of average accuracy. Subsequently, during the adaptation stage, the server synthesizes collective optimization trajectories to rectify local deviations, thereby yielding aligned client gradients. Finally, in the stage termed Anti-Bias Aggregation and Trajectory-Based Adaptation, the server obtains the global model via weighted aggregation and executes the final update using the aligned gradients and directional parameters derived in the preceding steps. For the first and second stages, the update procedure is implemented within a single training round t. Thus, for notational simplicity, we omit the superscript t in these two stages (Sections 3.2.1 and 3.2.2).

3.2.1. Evaluation of Anti-Bias Weights

The server evaluates each client using a publicly available probe dataset $\mathcal{D}_{\text{probe}}$ to compute anti-bias weights: the aggregation weight w_i, derived from the entropy of the prediction distribution to determine the contribution of the client during preliminary aggregation; and the bias penalty weight λ_i, which acts as a suppression coefficient for the client exhibiting excessive deviation from the average accuracy.

Aggregation Weight ($\boldsymbol{w_i}$). Prior to calculating the client aggregation weights, the server must first determine whether the model is biased. To achieve this, the server evaluates the model $W_i^{t,E}$ of each client on the probe dataset $\mathcal{D}_{\text{probe}}$ to obtain the softmax prediction probabilities. As illustrated in Figure 2, empirical observations indicate that unbiased models generally avoid overconfident misclassifications (e.g., Clients 1–3), resulting in diverse prediction outputs. In contrast, biased models tend to produce highly similar and overconfident predictions regardless of the input (e.g., Clients 4 and N). We term this phenomenon "model collapse". To quantify this phenomenon, we compute the average softmax probability distribution $\bar{\boldsymbol{p}}_i = \frac{1}{|\mathcal{D}_{\text{probe}}|}\sum_{d=1}^{|\mathcal{D}_{\text{probe}}|} \boldsymbol{p}_{i,d}$, where $\boldsymbol{p}_{i,d}$ is the probability distribution predicted by client i for the d-th data sample in $\mathcal{D}_{\text{probe}}$. We then define the concentration metric $\phi_i \in [0,1]$ based on the normalized entropy of $\bar{\boldsymbol{p}}_i$ (Algorithm 2, Lines 8–9):

$$\phi_i = 1 - \frac{-\bar{\boldsymbol{p}}_i \log(\bar{\boldsymbol{p}}_i)}{\log(C)}, \tag{2}$$

where C represents the number of classes. A higher ϕ_i signifies more severe model drift. Consequently, to suppress the influence of biased clients, we assign aggregation weight w_i inversely related to this concentration (Algorithm 2, Lines 9 and 14):

$$w_i = \frac{1-\phi_i}{\sum_{i=1}^{N}(1-\phi_i)}. \tag{3}$$

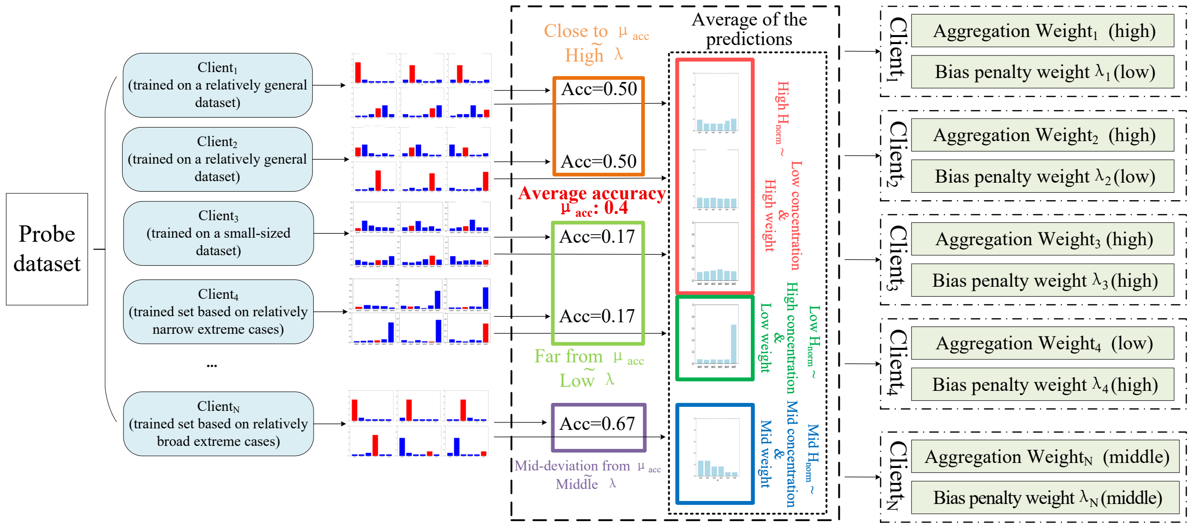

Probe dataset
Client₁ (trained on a relatively general dataset)
Client₂ (trained on a relatively general dataset)
Client₃ (trained on a small-sized dataset)
Client₄ (trained set based on relatively narrow extreme cases)
...
Client_N (trained set based on relatively broad extreme cases)
Close to μ acc
High λ
Acc=0.50
Acc=0.50
Average accuracy μ acc: 0.4
Acc=0.17
Acc=0.17
Far from μ acc
Low λ
Acc=0.67
Mid-deviation from μ acc
Middle λ
Average of the predictions
High H_norm ~ Low concentration & High weight
Low H_norm ~ High concentration & Low weight
Mid H_norm ~ Mid concentration & Mid weight
Client₁
Aggregation Weight₁ (high)
Bias penalty weight λ₁ (low)
Client₂
Aggregation Weight₂ (high)
Bias penalty weight λ₂ (low)
Client₃
Aggregation Weight₃ (high)
Bias penalty weight λ₃ (high)
Client₄
Aggregation Weight₄ (low)
Bias penalty weight λ₄ (high)
Client_N
Aggregation Weight_N (middle)
Bias penalty weight λ_N (middle)

Figure 2. Evaluation of anti-bias weights module. We use a 6-class probe dataset to visualize specific scenarios: typical bias (Client₁, Client₂), under-training (Client₃), and narrow or broad bias (Client₄, Client$_N$). For the bar charts, the horizontal and vertical axes represent data categories and prediction probability respectively, while blue and red bars indicate incorrect and correct predictions. The longer the bar, the higher the probability. This figure illustrates observed phenomena and remains independent of the experimental settings.

Algorithm 2 Proposed Two-Stage Aggregation Algorithm

```
Input: Initial model $W^0_{\text{global}}$, probe data $\mathcal{D}_{\text{probe}}$, number of rounds $T$, number of epochs $E$, number of clients $N$
Output: Updated global model $W^T_{\text{global}}$
1:  for each global round $t = 0$ to $T-1$ do
2:                                                        ▷ — Client-Side Operations —
3:      for each client $i = 1$ to $N$ in parallel do
4:          $W_i^{t,E}, P_i^t \leftarrow \textsc{ClientUpdate}(i, W^t_{\text{global}}, \mathcal{D}_i, E)$
5:      end for
6:                                                        ▷ — Server-Side Operations —
7:      for each client $i = 1$ to $N$ do
8:          $\bar{\boldsymbol{p}}_i, \text{Acc}_i^t \leftarrow \textsc{EvalProbe}(W_i^{t,E});\quad g_i^t \leftarrow \frac{1}{E}\sum_{e=0}^{E-1} P_i^t[e]$     ▷ Get Accuracy & Avg Gradient
9:          $\phi_i \leftarrow 1 - \frac{-\bar{\boldsymbol{p}}_i \log(\bar{\boldsymbol{p}}_i)}{\log(C)}$     ▷ Calculate Entropy
10:     end for
11:     $\mu^t_{\text{Acc}} \leftarrow \frac{1}{N}\sum_{i=1}^N \text{Acc}_i^t;\quad \bar{g}^t \leftarrow \frac{1}{N}\sum_{i=1}^N g_i^t$     ▷ Compute Means
12:     $W^{t+0.5}_{\text{global}} \leftarrow \mathbf{0};\quad U^t \leftarrow \mathbf{0}$     ▷ Initialize Global Model & Update Vector
13:     for each client $i = 1$ to $N$ do
14:         $w_i^t \leftarrow \frac{1-\phi_i}{\sum_{j=1}^N (1-\phi_j)}$
15:         $W^{t+0.5}_{\text{global}} \leftarrow W^{t+0.5}_{\text{global}} + w_i^t \cdot W_i^{t,E}$     ▷ Apply Weighted Aggregation
16:         $\lambda_i^t \leftarrow \exp(-\beta(\text{Acc}_i^t - \mu^t_{\text{Acc}})^2)$     ▷ Bias Penalty
17:         $\text{Sim}_i^t \leftarrow \frac{g_i^t \cdot \bar{g}^t}{\|g_i^t\| \cdot \|\bar{g}^t\|};\quad g_i'^t \leftarrow (1-\theta) \cdot g_i^t + \theta \cdot \bar{g}^t$     ▷ Similarity & Aligned Gradient
18:         if $(\phi_i \geq \tau_{\text{conc}})$ or $(\text{Sim}_i^t \leq \tau_{\text{sim}})$ then $D_i^t \leftarrow +1$
19:         else $D_i^t \leftarrow -1$
20:         end if
21:         $U^t \leftarrow U^t + (w_i^t \cdot \lambda_i^t \cdot D_i^t) \cdot g_i'^t$     ▷ Accumulate Directed Adaptation
22:     end for
23:     $W^{t+1}_{\text{global}} \leftarrow W^{t+0.5}_{\text{global}} + \eta \cdot U^t$     ▷ Update Global Model
24: end for
25: return $W^T_{\text{global}}$
```

Bias Penalty Weight ($\boldsymbol{\lambda_i}$). Relying solely on prediction concentration is insufficient, as it fails to detect under-trained models resulting from limited sample sizes (e.g., Client₃ in Figure 2). To address this, we develop bias penalty weights based on the deviation of the client accuracy Acc_i on $\mathcal{D}_{\text{probe}}$ relative to the cohort mean μ_{Acc} (Algorithm 2, Line 11). We distinguish between two outlier scenarios. A client with anomalously high accuracy ($\text{Acc}_i \gg \mu_{\text{Acc}}$) represents a relatively broad-biased case, implying its local model overfits to a specific, non-representative distribution. Conversely, extremely low accuracy ($\mathrm{Acc}_i \ll \mu_{\mathrm{Acc}}$) indicates a relatively narrow-biased case driven by severe drift or underfitting. We posit that clients performing near the cohort average ($\mathrm{Acc}_i \approx \mu_{\mathrm{Acc}}$) offer the most globally representative consensus. Consequently, we employ a Gaussian function to assign trust (Algorithm 2, Line 16):

$$\lambda_i \propto \exp(-\beta(\mathrm{Acc}_i - \mu_{\mathrm{Acc}})^2), \tag{4}$$

where $\beta \in [0, 1]$ is a hyperparameter controlling sharpness. This function acts as an adaptive filter throughout the training lifecycle. It dynamically assigns $\lambda_i \approx 1.0$ to the consensus cohort clustering around the mean, while effectively suppressing both broad and narrow biased outliers to ensure precise convergence.

3.2.2. Adaptation of Client Gradients

The server synthesizes uploaded optimization trajectories to derive a unified average gradient, which approximates the update magnitude of the complete dataset. By quantifying the similarity between individual trajectories and this average, the server rectifies local deviations to yield client-aligned gradients. Furthermore, integrating these similarity metrics with the aggregation weights from Section 3.2.1 enables the determination of the precise optimization direction, specifically whether to apply gradient descent or ascent for each client.

Client-Aligned Gradient (g'_i). As illustrated in the pipeline in Figure 3, the server first processes $P_i = [\Delta_i^0, \Delta_i^1, ..., \Delta_i^{E-1}]$. By calculating the average of all Δ vectors in the trajectory, it obtains the client average gradient $g_i = \frac{1}{E}\sum_{e=0}^{E-1} P_i[e]$ (Algorithm 2, Line 8). Here, g_i represents the "local model optimization intent" of client i during its local training. We posit that "model collapse" forces the model to take shortcuts, causing g_i to be heavily biased towards a specific subset of model parameters. To prevent any single g_i from exerting an excessive impact, we develop a regularization step. After obtaining all g_i, the server computes their arithmetic mean $\bar{g} = \frac{1}{N}\sum_{i=1}^{N} g_i$ (Algorithm 2, Line 11), which represents the average optimization trend of all clients and serves as an approximation of the correct client optimization trajectory. Using a hyperparameter $\theta \in [0, 1]$, we pull each g_i slightly towards $\bar{g}$ to obtain the aligned gradient g'_i (Algorithm 2, Line 17):

$$g'_i = (1 - \theta) \cdot g_i + \theta \cdot \bar{g}. \tag{5}$$

Here, θ denotes the update coefficient for the client-aligned gradient, while g'_i represents the actual adaptation vector employed in the final update.

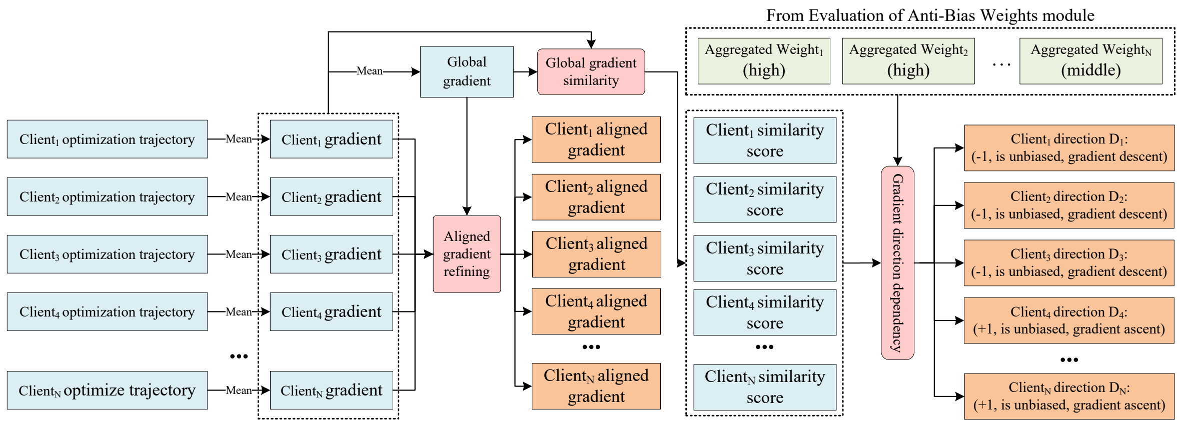

From Evaluation of Anti-Bias Weights module
Aggregated Weight$_1$ (high)
Aggregated Weight$_2$ (high)
...
Aggregated Weight$_N$ (middle)
Mean
Global gradient
Global gradient similarity
Client$_1$ optimization trajectory
Client$_2$ optimization trajectory
Client$_3$ optimization trajectory
Client$_4$ optimization trajectory
Client$_N$ optimize trajectory
Mean
Client$_1$ gradient
Client$_2$ gradient
Client$_3$ gradient
Client$_4$ gradient
Client$_N$ gradient
Aligned gradient refining
Client$_1$ aligned gradient
Client$_2$ aligned gradient
Client$_3$ aligned gradient
Client$_4$ aligned gradient
Client$_N$ aligned gradient
Client$_1$ similarity score
Client$_2$ similarity score
Client$_3$ similarity score
Client$_4$ similarity score
Client$_N$ similarity score
Gradient direction dependency
Client$_1$ direction D$_1$: (-1, is unbiased, gradient descent)
Client$_2$ direction D$_2$: (-1, is unbiased, gradient descent)
Client$_3$ direction D$_3$: (-1, is unbiased, gradient descent)
Client$_4$ direction D$_4$: (+1, is unbiased, gradient ascent)
Client$_N$ direction D$_N$: (+1, is unbiased, gradient ascent)

Figure 3. Adaptation of Client Gradients Module.

Direction of Adaptation (D_i^t). To determine the direction of adaptation (whether to apply gradient descent or ascent for each client), we compute the cosine similarity between g_i and $\bar{g}$, defining it as the similarity score Sim_i (Algorithm 2, Line 17):

$$\mathrm{Sim}_i = \frac{g_i \cdot \bar{g}}{\|g_i\| \cdot \|\bar{g}\|}. \tag{6}$$

The term Sim_i is a critical metric that is independent of ϕ_i. While ϕ_i (from Section 3.2.1) measures the result of drift (i.e., whether the model's final state is collapsed), Sim_i measures the process of drift (i.e., whether the path conflicted with the consensus). A low Sim_i value indicates that the optimization direction of the client is harmful, even if its data is not skewed enough to cause a high ϕ_i. Finally, we combine ϕ_i and Sim_i to classify client i as biased or unbiased. We set τ_{conc} and τ_{sim} as thresholds to determine whether the client-aligned gradient is biased.

Client i is labeled as biased if: $(\phi_i \geq \tau_{\text{conc}})$ or $(\text{Sim}_i \leq \tau_{\text{sim}})$. The direction of adaptation D_i is then defined as (Algorithm 2, Lines 18–19):

$$D_i = \begin{cases} +1, & \text{if client } i \text{ is biased;} \\ -1, & \text{if client } i \text{ is unbiased.} \end{cases} \tag{7}$$

3.2.3. Anti-Bias Aggregation and Trajectory-Based Adaptation

Synthesizing the preceding results, the server executes anti-bias aggregation and trajectory-based adaptation in two distinct phases. The pipeline of this module is shown in Figure 4. Phase I performs weighted aggregation utilizing the client weights derived in Section 3.2.1. Phase II refines this aggregated model by integrating the bias penalty weights and processed gradients from Section 3.2.2. To denote the iterative nature of the global training, we introduce the superscript t to represent the current global round index.

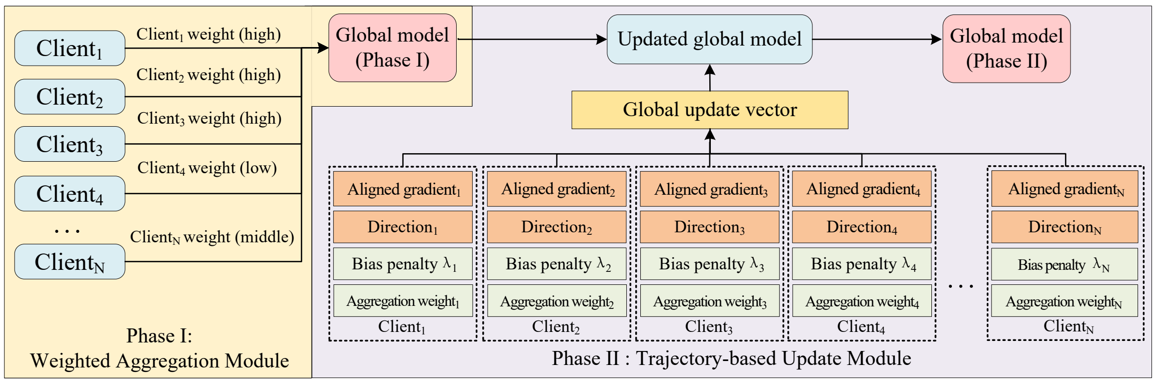


Figure 4. Anti-Bias Aggregation and Trajectory-Based Adaptation.

Phase I: Weighted Aggregation Module. The server first performs weighted aggregation on client models' parameters $W_i^{t,E}$ based on w_i^t from Section 3.2.1 to generate an intermediate aggregated model. Given that the global update protocol is divided into two distinct phases culminating in the final model W_{global}^{t+1}, we formally designate the intermediate model state generated after Phase I as $W_{\text{global}}^{t+0.5}$. This intermediate aggregation is computed as $W_{\text{global}}^{t+0.5} = \sum_{i=1}^{N} w_i^t \cdot W_i^{t,E}$ (Algorithm 2, Line 15), where w_i^t represents the aggregation weight for the i-th client, subject to the normalization constraint $\sum_{i=1}^{N} w_i^t = 1$.

Phase II: Trajectory-Based Update Module. Following the initial aggregation, the server performs a secondary trajectory-based update on the intermediate model $W_{\text{global}}^{t+0.5}$. We utilize the aggregation weights w_i, bias penalty weights λ_i, and directions of adaptation D_i^t derived in previous steps to construct an update vector. Specifically, the server computes the global update vector U^t by accumulating the weighted and directed aligned gradients $g_i'^t$ from all clients (Algorithm 2, Line 21):

$$U^t = \sum_{i=1}^{N} (w_i \cdot \lambda_i \cdot D_i^t) \cdot g_i'^t. \tag{8}$$

This formulation ensures that contributions from reliable clients are amplified while detrimental updates from biased clients are effectively reversed. Finally, the server applies U^t to the intermediate model with a global update rate $\eta \in [0, 1]$ to obtain the final model state W_{global}^{t+1} (Algorithm 2, Line 23):

$$W_{\text{global}}^{t+1} = W_{\text{global}}^{t+0.5} + \eta \cdot U^t. \tag{9}$$

3.3. Summary and Discussion

In summary, FedA^4 establishes a novel framework integrating anti-bias aggregation and trajectory-based adaptation. Distinct from existing methodologies that rely exclusively on the final state of client model parameters, FedA^4 leverages optimization trajectories to quantify client bias and rectify local deviations. This design explicitly addresses a critical limitation observed in empirical studies: weighted aggregation alone often converges toward but fails to surpass baseline performance. Consequently, FedA^4 moves beyond simple reweighting to adopt a granular optimization paradigm. By inducing global update dynamics directly from local trajectories, rather than relying on parameter regularization or synthetic data generation, the framework effectively approximates the training dynamics of the true global distribution, thereby preventing overfitting to Non-IID data.

4. Experiment

In this section, we conduct a comprehensive set of experiments to validate the effectiveness of FedA^4. We aim to answer the following primary questions: (1) Does FedA^4 outperform standard baselines under various Non-IID conditions and dataset complexities? (2) Is the performance gain of FedA^4 robust across different model architectures and different client numbers? (3) To what extent does each individual component contribute to the overall performance of FedA^4, and which component is the most critical?

4.1. Experimental Setup

Datasets. To comprehensively assess performance across diverse degrees of task complexity and image resolution, we conduct experiments on four widely recognized benchmark datasets: Fashion-MNIST [24], STL-10 [23], CIFAR-10, and CIFAR-100 [22]. CIFAR-10 [22] serves as the canonical benchmark for FL, containing 60,000 32 $\times$ 32 color images distributed across 10 classes. Fashion-MNIST [24], consisting of 70,000 28 $\times$ 28 grayscale images, is utilized to evaluate performance on lower-complexity and single-channel data. To examine robustness against label space complexity, we employ CIFAR-100 [22]; its 100 distinct classes significantly intensify the statistical heterogeneity arising from Non-IID partitioning. Finally, STL-10 [23] is leveraged to demonstrate the scalability and generalization capabilities of FedA^4 on tasks involving higher-resolution (96 $\times$ 96) input features.

To simulate a realistic heterogeneous environment, we partition the training data among $N = 10$ clients utilizing a Dirichlet distribution controlled by the hyperparameter α, which is greater than 0. This constitutes a standard methodology for modeling label distribution skew. The parameter α governs the degree of Non-IID heterogeneity. A smaller α (e.g., 0.1) yields a highly skewed and more challenging Non-IID distribution where clients may possess samples from only a limited number of classes. Conversely, a larger α (e.g., 1.0) result in a more balanced distribution resembling an IID setting.

Baseline Methods. We compare FedA^4 against three fundamental and widely adopted baseline methods: Avg (unweighted mean), FedAvg [1] and FedProx [13]. Our method involves both weighted aggregation and a corrective operation on the global model parameters. Therefore, comparing against Avg (unweighted Mean), FedAvg [1] (weighted fusion) and FedProx [13] (parameter constraint) is essential. We intentionally omitted more complex, recent SOTA methods (e.g., SCAFFOLD [14], FedGen [17], and FedLMG [18]) for a critical reason: stability and robustness. Our preliminary tests revealed that these advanced methods are often unstable; for instance, SCAFFOLD frequently suffered from training collapse, with test accuracy decreasing as training progressed. Furthermore, generative methods like FedGen and FedLMG failed to produce effective global data on CIFAR-10 and CIFAR-100 in our setup, preventing the global model from optimizing effectively. We thus focus our comparison on these robust and universally applicable baselines.

Implementation Details. Unless otherwise specified, all experiments utilize a *StandardCNN* architecture consisting of six layers as the client model. We configure the experimental setup with $N = 10$ clients, $E = 3$ local epochs, and a total of $T = 50$ global rounds. Local training employs the Adam optimizer with a learning rate of 10^{-3} and a batch size of 64. For the FedA^4 framework, all hyperparameters are fine-tuned and remain fixed across all evaluated datasets. We provide general guidance on setting these hyperparameters. By default, we suggest $\beta = 1.0$, $\eta = 0.01$, $\theta = 0.9$, $\tau_{conc} = 0.3$, and $\tau_{sim} = 0.2$ for stable convergence performance. For scenarios involving extreme non-IID data distributions, we recommend increasing β and η while decreasing τ_{conc}. If training instability arises, θ should be increased and τ_{sim} reduced. Regarding adjustment scales, β should be tuned on the order of 10^0, θ, τ_{conc}, and τ_{sim} should be adjusted on the order of 10^{-1}, and η is best modified in increments of 10^{-2}. We construct the probe dataset $\mathcal{D}_{\text{probe}}$ by sampling $K = 1$ data sample for each class $c \in \mathcal{C}$ from the global training set. We explicitly ensure that the probe dataset and local client datasets remain disjoint, satisfying the condition $\mathcal{D}_{\text{probe}} \cap \mathcal{D}_i = \emptyset$ for all i.

4.2. Comparison with Baseline Methods

We present the primary experimental results comparing the final global model test accuracy of FedA^4 against baseline methods across all four datasets and various heterogeneity settings. The quantitative data in Table 1 substantiates the efficacy of the proposed framework. Overall, FedA^4 exhibits consistent superiority, outperforming competing approaches across the evaluated scenarios.

Table 1. Test accuracy comparison on benchmark datasets under varying Non-IID settings. The bold shows the best result or accuracy increase.

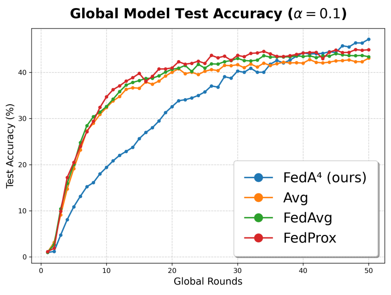
	CIFAR-10						**CIFAR-100**					
	0.1	**0.3**	**0.5**	**0.7**	**1.0**	**Avg**	**0.1**	**0.3**	**0.5**	**0.7**	**1.0**	**Avg**
Avg	73.98	83.63	84.37	84.57	85.36	82.38	49.18	52.66	53.91	54.10	55.25	53.02
FedAvg	74.45	82.78	84.50	**85.19**	85.67	82.52	49.50	52.19	53.93	53.97	54.94	52.91
FedProx	70.86	81.10	83.59	83.97	85.30	80.96	51.60	55.73	57.11	57.97	58.89	56.26
FedA^4(ours)	**77.24** **(+2.79)**	**84.13** **(+0.50)**	**84.78** **(+0.28)**	85.09 (-0.10)	**85.79** **(+0.12)**	**83.41** **(+0.89)**	**52.16** **(+0.56)**	**60.70** **(+4.97)**	**63.03** **(+5.92)**	**63.93** **(+5.96)**	**64.54** **(+5.65)**	**60.87** **(+4.61)**
	Fashion-MNIST						**STL-10**					
	0.1	**0.3**	**0.5**	**0.7**	**1.0**	**Avg**	**0.1**	**0.3**	**0.5**	**0.7**	**1.0**	**Avg**
Avg	88.72	**93.38**	**93.74**	93.33	93.81	92.60	43.43	57.45	61.19	60.75	61.84	56.93
FedAvg	88.72	93.34	93.71	93.39	93.94	92.62	51.41	56.61	60.57	61.06	61.83	58.30
FedProx	89.11	92.99	93.34	93.28	**94.21**	92.59	52.68	57.53	59.48	60.95	62.84	58.70
FedA^4(ours)	**90.27** **(+1.16)**	93.33 (-0.04)	93.47 (-0.27)	**93.80** **(+0.41)**	94.11 (-0.10)	**93.00** **(+0.38)**	**56.32** **(+3.64)**	**59.16** **(+1.63)**	**63.02** **(+1.83)**	**63.32** **(+2.13)**	**64.65** **(+1.81)**	**61.29** **(+2.59)**

We initiate our evaluation using the CIFAR-10 dataset. As detailed in the top-left section of Table 1, FedA^4 outperforms competing baselines across nearly all data partition settings defined by the Dirichlet distribution. To verify applicability to lower-complexity grayscale classification tasks, we examine the Fashion-MNIST results presented in the bottom-left of Table 1. These findings highlight substantial improvements, particularly under conditions of severe statistical heterogeneity ($\alpha = 0.1$), while maintaining the highest average accuracy across all α settings compared to alternative schemes. Following the assessment on CIFAR-10 and lower-complexity datasets, we extend our evaluation to more challenging scenarios. To validate efficacy in environments with expanded label spaces, we utilize CIFAR-100. As shown in the top-right of Table 1, FedA^4 consistently surpasses baseline methods across all α values, with performance gains becoming particularly pronounced when $\alpha > 0.1$. Similarly, to demonstrate robustness in handling high-dimensional inputs, we employ the STL-10 dataset. The results presented in the bottom-right of Table 1 confirm that FedA^4 maintains superior performance across all distribution settings. Collectively, these findings demonstrate that FedA^4 achieves state-of-the-art performance across diverse task complexities and varying degrees of Non-IID heterogeneity.

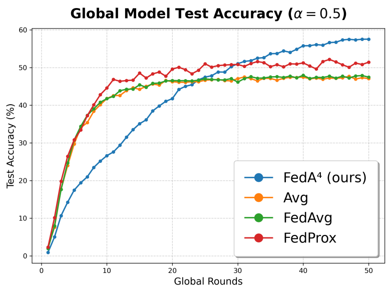
4.3. Scalability of Model Architecture

To verify that the performance enhancements of FedA^4 are architecture-independent and not restricted to the specific six-layer CNN, we conducted supplementary experiments on the CIFAR-100 dataset using models of varying complexity. Specifically, we evaluated a shallow three-layer CNN and a deeper ResNet-18 architecture. The results, presented in Figure 5, indicate that FedA^4 consistently achieves higher final accuracy than all baseline methods. This superiority persists regardless of model complexity (three-layer, six-layer, or ResNet-18) and across diverse α values. These findings strongly suggest that our method constitutes a generalizable framework for FL rather than a solution tailored to a specific architecture.

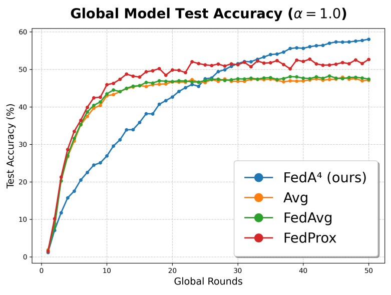
As illustrated in Figure 5, the global model accuracy curves exhibit a distinct characteristic extending beyond the previously observed trend regarding performance gaps at various Dirichlet α values. Specifically, the curves demonstrate the capacity of FedA^4 to escape local optima, a behavior visually manifested as a slower initial convergence rate followed by superior final accuracy. As discussed in Section 3, this phenomenon arises because baseline methods remain susceptible to the detrimental influence of biased clients, frequently causing the global model to converge prematurely to local optima associated with skewed distributions. In contrast, FedA^4 leverages intrinsic evaluation and adaptation mechanisms to actively identify and rectify deviations from the global optimization path.



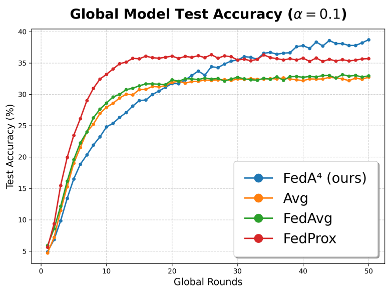
(a) ResNet-18, $\alpha = 0.1$



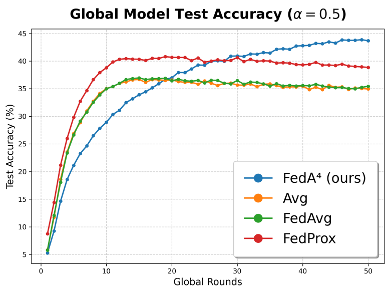
(b) ResNet-18, $\alpha = 0.5$



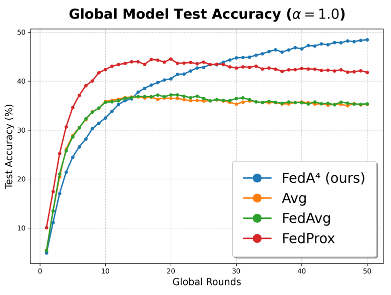
(c) ResNet-18, $\alpha = 1.0$



(d) 3-Layer CNN, $\alpha = 0.1$



(e) 3-Layer CNN, $\alpha = 0.5$



(f) 3-Layer CNN, $\alpha = 1.0$

Figure 5. Scalability of model architecture when models are trained on CIFAR-100 [22]. The top row illustrates the test accuracy comparison between FedA^4 and baselines using the ResNet-18 model under Dirichlet distributions of $\alpha \in \{0.1, 0.5, 1.0\}$. The bottom row presents the comparison using a 3-layer CNN model under the same heterogeneity settings.

4.4. Robustness to Different Client Numbers

As evidenced in Table 2, variations in the number of participating clients exert negligible impact on the performance of the proposed method. FedA^4 maintains high accuracy in almost all scenarios regardless of whether the client cohort size is set to 5 or 20. This stability across varying scales underscores the robustness of the framework, confirming that the efficacy of the bias adaptation mechanism remains independent of the specific number of participants.

Table 2. Performance comparison under different client numbers and heterogeneity settings. The bold shows the best result or accuracy increase.

	$\alpha = 0.1$			$\alpha = 0.3$			$\alpha = 0.7$			$\alpha = 1.0$		
	5 Clients	**10 Clients**	**20 Clients**	**5 Clients**	**10 Clients**	**20 Clients**	**5 Clients**	**10 Clients**	**20 Clients**	**5 Clients**	**10 Clients**	**20 Clients**
Avg	71.83	73.98	74.49	81.02	83.63	81.73	84.69	84.57	83.39	85.83	85.36	84.30
FedAvg	71.68	74.45	75.79	78.92	82.78	**82.20**	84.24	**85.19**	83.70	85.55	85.67	84.32
FedProx	73.05	70.86	71.95	81.60	81.10	81.54	84.55	83.97	83.31	85.91	85.30	84.29
FedA^4	**74.11** **(+1.06)**	**77.24** **(+2.79)**	**75.86** **(+0.07)**	**81.95** **(+0.35)**	**84.13** **(+0.50)**	81.45 (-0.75)	**85.25** **(+0.56)**	85.09 (−0.10)	**83.77** **(+0.07)**	**86.18** **(+0.27)**	**85.79** **(+0.12)**	**84.58** **(+0.27)**

4.5. Ablation Study

We validate the proposed FedA^4 framework by performing an ablation study on its two core components: the weighted aggregation module (denoted as *Agg. Weights*) and the trajectory-based update module (denoted as *Traj. Update*). Notably, the simultaneous exclusion of both modules reduces the system to the FedAvg baseline.

As presented in Table 3, the complete FedA^4 architecture achieves superior performance across all evaluated scenarios. Furthermore, each component individually contributes to accuracy improvements over the baseline, confirming that the two modules are complementary. Crucially, the *Traj. Update* module provides a more significant performance enhancement than the *Agg. Weights* module alone. This empirical finding corroborates our design hypothesis: while reweighting offers passive mitigation against biased clients, actively rectifying the optimization direction via client trajectories addresses the underlying cause of drift more effectively. In the case of Dirichlet $\alpha = 1.0$, using the anti-bias aggregation method alone may not yield an accuracy increase. This is because under a near-IID data distribution ($\alpha = 1.0$), the anti-bias aggregation reduces to weighted average aggregation, achieving similar accuracy (85.67 vs. 85.60).

Table 3. Ablation study on the core components of FedA^4. The bold shows the best result or accuracy increase.

Dirichlet $\alpha = 0.1$			**Dirichlet $\alpha = 1.0$**		
Agg. Weights	**Traj. Update**	**Test Acc**	**Agg. Weights**	**Traj. Update**	**Test Acc**
×	×	74.45	×	×	85.67
√	×	74.65 (+0.20)	√	×	85.60 (−0.07)
×	√	76.70 (+2.25)	×	√	85.75 (+0.08)
√	√	**77.24 (+2.79)**	√	√	**85.79 (+0.12)**

5. Conclusions and Future Works

In this paper, we presented FedA^4, a novel framework incorporating Anti-Bias Aggregation and Trajectory-Based Adaptation mechanisms designed to address the persistent challenge of client drift in Non-IID FL environment. Departing from conventional approaches that rely solely on aggregation weight adjustment or final model parameter adaptation, FedA^4 leverages client optimization trajectories to actively diagnose and rectify the directional bias of local models. Extensive empirical evaluations across diverse datasets, client scales, and model architectures demonstrate that our approach consistently outperforms baseline methods, exhibiting remarkable robustness against varying degrees of statistical heterogeneity. Notwithstanding these contributions, we acknowledge that the current framework entails architectural complexity, relies partially on heuristic logic for gradient adjustment, and necessitates a public probe dataset. Consequently, future work will focus on establishing a rigorous mathematical framework to derive theoretically sound global gradients, streamlining the architecture to enhance generalization, and integrating generative models to synthesize probe data, thereby eliminating external dependencies and reducing model complexity. Moreover, the FL security should also be considered, including malicious client attacks and gradient inversion attacks. Ultimately, FedA^4 provides a potent and privacy-preserving paradigm for constructing unbiased global models. The novel application of optimization trajectories not only validates the superiority of the framework but also illuminates new research avenues within the field of FL.

Author Contributions

G.Z.: Conceptualization, investigation, writing, and revision. J.H.: Conceptualization, investigation, writing, and revision. Z.Y.: Conceptualization, investigation, writing, and revision. D.W.: Conceptualization, investigation, writing, and revision. All authors have read and agreed to the published version of the manuscript.

Funding

This paper is partially supported by Hong Kong Research Grants Council grants #11203523, #11209425, C1042-23GF, and Hong Kong Innovation and Technology Commission grant MHP/034/22.

Data Availability Statement

Not applicable.

Conflicts of Interest

The authors declare no conflict of interest. Given the role as Editor-in-Chief, Dapeng Oliver Wu had no involvement in the peer review of this paper and had no access to information regarding its peer-review process. Full responsibility for the editorial process of this paper was delegated to another editor of the journal.

Use of AI and AI-Assisted Technologies

During the preparation of this work, the authors used ChatGPT to check the grammar. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

1. McMahan, B.; Moore, E.; Ramage, D.; et al. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.

2. Konečnỳ, J.; McMahan, H.B.; Ramage, D.; et al. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv* **2016**, arXiv:1610.02527.
3. Li, T.; Sahu, A.K.; Talwalkar, A.; et al. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60.
4. Zhao, Y.; Li, M.; Lai, L.; et al. Federated learning with non-iid data. *arXiv* **2018**, arXiv:1806.00582.
5. Li, T.; Sanjabi, M.; Beirami, A.; et al. Fair resource allocation in federated learning. *arXiv* **2019**, arXiv:1905.10497.
6. Ji, S.; Pan, S.; Long, G.; et al. Learning private neural language modeling with attentive aggregation. In Proceedings of the 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 14–19 July 2019; pp. 1–8.
7. T Dinh, C.; Tran, N.; Nguyen, J. Personalized federated learning with moreau envelopes. In Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020.
8. Fu, S.; Yang, Z.; Hu, C.; et al. Personalized federated learning with contrastive momentum. *IEEE Trans. Big Data* **2024**, *11*, 2184–2194.
9. Blanchard, P.; El Mhamdi, E.M.; Guerraoui, R.; et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In Proceedings of the Advances in Neural Information Processing Systems 30, Long Beach, CA, USA, 4–9 December 2017.
10. Wang, H.; Yurochkin, M.; Sun, Y.; et al. Federated learning with matched averaging. *arXiv* **2020**, arXiv:2002.06440.
11. Yurochkin, M.; Agarwal, M.; Ghosh, S.; et al. Bayesian nonparametric federated learning of neural networks. In Proceedings of the 36 th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 7252–7261.
12. Lin, T.; Kong, L.; Stich, S.U.; et al. Ensemble Distillation for Robust Model Fusion in Federated Learning. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, CA, USA, 6–12 December 2020.
13. Li, T.; Sahu, A.K.; Zaheer, M.; et al. Federated optimization in heterogeneous networks. In Proceedings of the Third Conference on Machine Learning and Systems MLSys 2020, Austin, TX, USA, 2–4 March 2020.
14. Karimireddy, S.P.; Kale, S.; Mohri, M.; et al. Scaffold: Stochastic controlled averaging for federated learning. In Proceedings of the 37th International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 5132–5143.
15. Acar, D.A.E.; Zhao, Y.; Navarro, R.M.; et al. Federated learning based on dynamic regularization. *arXiv* **2021**, arXiv:2111.04263.
16. Gao, L.; Fu, H.; Li, L.; et al. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 10112–10121.
17. Zhu, Z.; Hong, J.; Zhou, J. Data-free knowledge distillation for heterogeneous federated learning. In Proceedings of the 38th International Conference on Machine Learning, Virtual, 18–24 July 2021; pp. 12878–12889.
18. Yang, M.; Su, S.; Li, B.; et al. One-Shot Heterogeneous Federated Learning with Local Model-Guided Diffusion Models. *arXiv* **2025**, arXiv:cs.CV/2311.08870.
19. Goodfellow, I.J.; Pouget-Abadie, J.; Mirza, M.; et al. Generative adversarial nets. In Proceedings of the NIPS'14: Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014
20. Zhu, L.; Liu, Z.; Han, S. Deep leakage from gradients. In Proceedings of the NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019.
21. Zhao, B.; Mopuri, K.R.; Bilen, H. Dataset condensation with gradient matching. *arXiv* **2020**, arXiv:2006.05929.
22. Krizhevsky, A. *Learning Multiple Layers of Features from Tiny Images*; University of Toronto: Toronto, ON, Canada, 2009.
23. Coates, A.; Ng, A.; Lee, H. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 11–13 April 2011; Volume 15, pp. 215–223.
24. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
25. Reddi, S.; Charles, Z.; Zaheer, M.; et al. Adaptive federated optimization. *arXiv* **2020**, arXiv:2003.00295.
26. Wang, R.; Chen, Y. Adaptive Model Aggregation in Federated Learning Based on Model Accuracy. *IEEE Wireless Commun.* **2024**, *31*, 200–206.
27. Pillutla, K.; Kakade, S.M.; Harchaoui, Z. Robust aggregation for federated learning. *IEEE Trans. Signal Process.* **2022**, *70*, 1142–1154.
28. Li, Q.; He, B.; Song, D. Model-contrastive federated learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 10713–10722.
29. Rasouli, M.; Sun, T.; Rajagopal, R. Fedgan: Federated generative adversarial networks for distributed data. *arXiv* **2020**, arXiv:2006.07228.