Efficient Asynchronous Federated Learning with Prospective Momentum Aggregation and Fine-Grained Correction

Yu Zang[1], **Zhe Xue**[1*], **Shilong Ou**[1], **Lingyang Chu**[2], **Junping Du**[1], **Yunfei Long**[1]
[1]Beijing University of Posts and Telecommunications, Beijing, China
[2]McMaster University, Hamilton, Canada
zyzy@bupt.edu.cn, xuezhe@bupt.edu.cn, osl@bupt.edu.cn, chul9@mcmaster.ca, junpingd@bupt.edu.cn, longyunfei@bupt.edu.cn

Abstract

Asynchronous federated learning (AFL) is a distributed machine learning technique that allows multiple devices to collaboratively train deep learning models without sharing local data. However, AFL suffers from low efficiency due to poor client model training quality and slow server model convergence speed, which are a result of the heterogeneous nature of both data and devices. To address these issues, we propose Efficient Asynchronous Federated Learning with Prospective Momentum Aggregation and Fine-Grained Correction (FedAC). Our framework consists of three key components. The first component is client weight evaluation based on temporal gradient, which evaluates the client weight based on the similarity between the client and server update directions. The second component is adaptive server update with prospective weighted momentum, which uses an asynchronous buffered update strategy and a prospective weighted momentum with adaptive learning rate to update the global model in server. The last component is client update with fine-grained gradient correction, which introduces a fine-grained gradient correction term to mitigate the client drift and correct the client stochastic gradient. We conduct experiments on real and synthetic datasets, and compare with existing federated learning methods. Experimental results demonstrate effective improvements in model training efficiency and AFL performance by our framework.

Introduction

In today's data-sensitive world, where privacy is paramount, federated learning (FL) emerges as a promising paradigm. It enables multiple devices to train deep learning models in parallel without sharing local data (Li et al. 2020b; Kairouz et al. 2021). This innovative approach not only safeguards user privacy but also presents unprecedented opportunities for achieving data-driven intelligence. In FL, each device (also called a client) trains a local model on its own data and periodically communicates with a central server to aggregate the local models into a global model. The server then broadcasts the updated global model to the clients for the next round of training. This process is repeated until the global model converges or meets some predefined criteria.

[*]Corresponding author.

Many existing FL methods use a synchronous communication scheme, which requires the participation of a certain percentage of clients in each round of aggregation (McMahan et al. 2017; Zang et al. 2023; Long et al. 2023; Guan et al. 2021; Li, Li, and Xue 2022). This may cause high communication cost and low scalability, especially when the number of clients is large or when the clients have different computing power or network conditions. Furthermore, synchronous federated learning (SFL) might not be well-suited for application scenarios involving heterogeneous devices. These scenarios often display dynamic attributes, including variations in device availability and device capabilities (Li et al. 2020a; Zhuo and Li 2021; Zhuang et al. 2020). Such attributes can lead to imbalanced and inefficient communication between clients and the server, ultimately diminishing the overall performance and robustness of FL models. To overcome these limitations, AFL has been proposed as an alternative communication scheme, which allows clients to communicate with the server at their own pace without waiting for others (Xie, Koyejo, and Gupta 2019). AFL can enhance the flexibility and scalability of FL by adapting to various client conditions and reducing communication bottlenecks, making it better suited to meet the challenges of practical application scenarios.

While AFL has shown promising results in enhancing the flexibility and scalability of FL, it still faces some challenges in achieving high efficiency and performance, particularly in dealing with data heterogeneity and device heterogeneity (Xu et al. 2021). In order to improve the efficiency and performance of AFL, various approaches have been proposed (Zhang et al. 2021; Li and Wang 2022; He et al. 2022; Koloskova, Stich, and Jaggi 2022; Jiang et al. 2022). These predominantly target three key facets: mitigated client staleness, accelerated server convergence speed, and improved client training quality. First, mitigated client staleness aims to assign proper weights to different clients based on their contribution to the global model. This can help alleviate the adverse impact of device heterogeneity on AFL and enhance the efficiency of updating the global model (Chen, Sun, and Jin 2019; Shi et al. 2020; Zhou et al. 2021; Wang et al. 2022b). Second, the goal of accelerated server convergence speed is to employ suitable aggregation strategies, which helps mitigate the impact of straggler devices. This reduces communication rounds during model convergence, thereby enhancing the efficiency of AFL (Wu et al. 2020; Nguyen et al. 2022a; So et al. 2021; Shi et al. 2020; Wang et al. 2022a). Lastly, the objective of improved client training quality is to mitigate the influence of client drift towards local optima updates caused by data heterogeneity, which can improve the overall performance of the global model. Existing methods have attempted to improve client training quality by adding aggregation constraints (Chai et al. 2020; Chen et al. 2020), clustered FL (Sattler, Müller, and Samek 2020; Lee et al. 2020), adjusting client training rounds (Wang et al. 2022b), and other approaches.

However, despite the advancements in AFL algorithms, they still exhibit certain limitations. First, the mitigated client staleness algorithm takes into account model staleness solely from a temporal perspective by assigning higher weights to clients with more frequent updates. However, focusing only on temporal does not capture the consistency between client and global pseudo-gradient directions (Kingma and Ba 2014). This can result in sub-optimal updates and slow convergence of the global model. Second, while existing methods (Nguyen et al. 2022b; Wu et al. 2020) aiming to accelerate server convergence speed utilize buffer aggregation strategies to compute the global pseudo-gradient, they have not delved deeply into harnessing the advanced optimization techniques like momentum algorithms, as extensively demonstrated in SFL (Reddi et al. 2021; Hsu, Qi, and Brown 2019). This oversight may result in protracted training durations and sub-optimal global models. Lastly, the existing improved client training quality algorithms mainly focus on coarse-grained adjustments over rounds or epochs to mitigate client drift induced by data heterogeneity. However, these methods lack necessary fine-grained supervision over client stochastic gradients at each training step. This oversight can infuse biases and inaccuracies into client updates, ultimately undermining the integrity and accuracy of the global model.

To address the aforementioned issues, we propose efficient AFL with prospective momentum aggregation and fine-grained correction, which includes three components: 1) Client weight evaluation based on temporal gradient: We evaluate the client weight by measuring the consistency between the pseudo-gradients of the client model and the global model during the time interval in which the client performs its local update. We use this as the weight for each client during server-buffered aggregation; 2) Adaptive server update with prospective weighted momentum: We aggregate the local pseudo-gradients of all clients in buffer to obtain the global prospective weighted momentum used to update the global model with adaptive learning rate; 3) Client update with fine-grained gradient correction: To address client drift caused by heterogeneous data, we calculate a fine-grained correction term using client weight and client gradient information in buffer, and combine it with a local correction term to correct the gradients of each step during client update. Overall, our algorithm accelerates the convergence of the global model while mitigating client drift caused by heterogeneous data and device, improving both the efficiency and performance of AFL. The main contributions of this work can be summarized as:

- We propose a client weight evaluation method based on temporal gradient, which measures the client weight by assessing the consistency between the local and global pseudo-gradient directions, resulting in improving efficiency and performance of AFL and alleviating the adverse impact of device heterogeneity.
- We propose an adaptive server update with prospective weighted momentum. By leveraging client weight and buffer aggregation, we calculate a prospective weighted momentum, facilitating adaptive server updates. This enhances the convergence speed of the global model and AFL efficiency.
- We propose the notion of client updates incorporating fine-grained gradient correction. By leveraging temporal client weight, our approach computes a fine-grained gradient correction for client updates, rectifying local stochastic gradients, countering data heterogeneity, and improving both the performance and efficacy of AFL.

Related Work

Synchronous federated learning. In recent years, due to the issue of data privacy, FL has gained increasing attention. How to learn a global model quickly and effectively under non-independent and identically distributed (non-IID) data, and improve the efficiency of FL, is a question that everyone is concerned about. Many works have attempted to improve the efficiency of FL. Some (Acar et al. 2021; Li et al. 2020c; Gao et al. 2022) have tried to design corrections for client training from the perspective of mitigating bias to improve the efficiency of FL, while others (Hsu, Qi, and Brown 2019; Reddi et al. 2021) have introduced more effective aggregation strategies from the perspective of server aggregation to improve the efficiency of FL.

Asynchronous federated learning. AFL is often more suitable for real-world applications than SFL due to the straggler effect caused by device heterogeneity (Li et al. 2020b; Xu et al. 2021; Kairouz et al. 2021). AFL mainly faces the problem of outdated local models due to device heterogeneity and low-quality local models due to data heterogeneity. Various methods (Wu et al. 2020; Nguyen et al. 2022a; So et al. 2021) have been proposed to alleviate device heterogeneity by using different model weight aggregation and client selection schemes, and some methods (Chai et al. 2020; Chen et al. 2020) have attempted to design constraint terms or use clustered FL (Sattler, Müller, and Samek 2020; Lee et al. 2020) to mitigate client bias in asynchronous scenarios.

Methodology

Preliminary

In this section, we introduce the basic settings of FL and the buffered aggregation scheme that we use.

The objective function for FL is as follows:

$$\min_{x} F(x) = \sum\nolimits_{i \in N} \frac{|D_i|}{|D|} F_i(x), \tag{1}$$

where N is the number of clients, x are the model parameters of client i, $F_i(x)$ is client i's local objective function, which describes how model parameters conforms to clients i's local dataset, D_i is the number of data samples in client i, and $|D| = \sum_{i \in N} |D_i|$ is the total number of data samples across all clients.

A common algorithm for SFL is Federated Averaging (FedAvg) (McMahan et al. 2017). In each communication round, a subset of clients S are selected to participate in the model training. On each client i, the client first downloads the current global model parameters x^{t-1} from the server in communication round t. Then, it trains the local model by running K steps of stochastic gradient descent (SGD) on the local dataset:

$$x_{i,k+1}^{t-1} = x_{i,k}^{t-1} - \eta_l g_i(x_{i,k}^{t-1}), \tag{2}$$

where η_l is the client learning rate, k is current step, $g_i(x_{i,k}^{t-1})$ is the local stochastic gradient.

In this paper, we adopt the asynchronous buffered aggregation framework (FedBuff) (Nguyen et al. 2022b) as the basis for our framework. In FedBuff, clients participate in training and communication with the server asynchronously. However, unlike other asynchronous methods, the server does not immediately aggregate the model after receiving updates from individual clients. Instead, the server waits for a buffer to contain updates from a group of B clients before performing the aggregation process, as described in Eq.(3):

$$x^t = x^{t-1} + \eta_g\left(\frac{\gamma \times s(t-\tau)}{|B|} \sum_{i \in B} (x_{i,K}^{\tau} - x^{\tau})\right), \tag{3}$$

where B is clients set in the buffer that stores the client updates, x^{τ} represents the server model downloaded by the client in round τ, η_g is global learning rate, $(x_{i,K}^{\tau} - x^{\tau})$ is the client update pseudo-gradient (Reddi et al. 2021), γ is a hyperparameter, $K = E * |Di|/batchsize$. The term $\gamma \times s(t-\tau)$ represents a function mapping same as (Xie, Koyejo, and Gupta 2019) related to the staleness of the clients' updates. In FedBuff, $s(t-\tau)$ is defined as $(1 + (t-\tau))^{-0.5}$.

It is worth noting that the buffer aggregation method, although similar to SFL in terms of aggregation, is fundamentally different. In the asynchronous buffer aggregation algorithm, clients do not need to wait for the server to broadcast the global model parameters or for specific clients to complete their updates before performing global updates. All clients update asynchronously in parallel, with the buffer solely influencing the server's update frequency, which is independent of the clients' parallel updates.

Client Weight Evaluation Based on Temporal Gradient

In traditional AFL (Xie, Koyejo, and Gupta 2019), client weights depend on client time staleness, which is calculated as the time difference between client's last communication at round τ and its return of the model at round t, as shown in Eq. (2). The traditional client weight evaluation takes into account model staleness solely from a temporal perspective by assigning higher weights to clients with more frequent updates. However, focusing only on temporal does not capture the consistency between client and global pseudo-gradient directions. This can result in sub-optimal performance when updating the global model.

To overcome this, we propose using cosine similarity between the local and global pseudo-gradients within the time span $t - \tau$ to determine client weights. The equation for calculating the weight of the client is as follows:

$$r_i^t = \frac{(x^{t-1} - x^{\tau})^T (x_{i,K}^{\tau} - x^{\tau})}{||x^{t-1} - x^{\tau}|| \cdot ||x_{i,K}^{\tau} - x^{\tau}||}, \tag{4}$$

where the difference $x^{t-1} - x^{\tau}$ represents global update pseudo-gradient, $x_{i,K}^{\tau} - x^{\tau}$ represents client update pseudo-gradient trained in K steps. By calculating the cosine similarity between these two pseudo-gradient, we can obtain a measure of the pseudo-gradient contributed by client i at time t.

After the number of client pseudo-gradients stored in the buffer reaches the specified quantity B, we normalize the weights of each client in the buffer, as shown below:

$$w_i^t = \frac{r_i^t}{\sum_{j \in B} r_j^t}, \tag{5}$$

where w_i^t represents the weight of client i in the buffer aggregation at round t. By normalizing, we map the client weights in the buffer to the range $[0, 1]$, which facilitates subsequent aggregation processing. With the client weight based on temporal gradient w_i^t in hand, we apply them to server aggregation and client update correction, as described in server and client update, respectively.

We evaluate the client weight using the temporal gradient by measuring the similarity between the pseudo-gradients of the client model and the global model during the time interval in which the client performs its local update. Our approach allows the client weight to capture both client staleness and the quality of their pseudo-gradients, thereby enhancing the efficiency and performance of AFL.

Adaptive Server Update with Prospective Weighted Momentum

Under the influence of data and device heterogeneity, client pseudo-gradients tend to be biased and staleness. Using these client pseudo-gradients to calculate momentum during model aggregation can exacerbate global model oscillations. Consequently, the incorporation of momentum for server updates in AFL methods is infrequent. However, the introduction of buffer aggregation provides relief by mitigating the disruptions caused by slower devices. Furthermore, by utilizing client weight based on temporal gradient, we can assign higher weights to clients with greater consistency in their pseudo-gradients with the global updates. This approach helps mitigate the bias and staleness of the global pseudo-gradients.

Upon buffer saturation, we compute global pseudo-gradients by leveraging client pseudo-gradients stored in the buffer, akin to the approach employed in FedBuff (Nguyen et al. 2022b). However, we differentiate by incorporating client weight based on temporal gradient. The procedure unfolds as follows:

$$g(x_i^{\tau}) = x_{i,K}^{\tau} - x^{\tau}, \tag{6}$$

$$g(x^t) = \sum\nolimits_{i \in B} w_i^t g(x_i^\tau). \tag{7}$$

By combining our client weight with asynchronous buffer aggregation, we can mitigate the impact of data and device heterogeneity on global pseudo-gradients.

Subsequently, we calculate the global momentum in a manner similar to FedAdam (Reddi et al. 2021) in SFL:

$$m^t = \beta_1 m^{t-1} + (1-\beta_1) g(x^{t-1}). \tag{8}$$

Next, we amalgamate Nesterov's accelerated gradient (NAG) (Nesterov 1983) and adaptive moment estimation (Adam) (Kingma and Ba 2014) to confer the global momentum with a forward-looking perspective and couple with adaptive global learning rate adjustment as Nadam (Dozat 2016), enhances the stability of server aggregation. The update equations for the server model is shown below:

$$v^t = \beta_2 v^{t-1} + (1-\beta_2) g^2(x^{t-1}), \tag{9}$$

$$\hat{m^t} = \beta_1 m^t + (1-\beta_1) g(x^{t-1}), \tag{10}$$

$$x^t = x^{t-1} + \frac{\eta_g \hat{m^t}}{\sqrt{v^t} + \epsilon}, \tag{11}$$

where ϵ is a small constant to avoid division by zero, β_1 and β_2 are hyperparameters.

Our method, adaptive server update with prospective weighted momentum, combines client weight based on temporal gradient with asynchronous buffer aggregation, applying momentum optimization to AFL. Additionally, during global model updates, we employ the advanced Nadam algorithm to expedite server aggregation, thereby enhancing the efficiency of AFL.

Client Update with Fine-Grained Gradient Correction

In FL, data heterogeneity often causes client model training to optimize towards local optima rather than global optima, a phenomenon known as *client drift*. In the context of AFL, the issue of client drift is worsened by the temporal lag of the global model. This lag leads to a lower quality of the global model when suboptimally trained client models are aggregated. However, the existing improved client training quality algorithms mainly focus on coarse-grained adjustments over rounds or epochs to mitigate client drift induced by data heterogeneity.

To address this issue, we propose a client update with fine-grained gradient correction algorithm that uses a temporal gradient correction term to correct the local stochastic gradient. The client update rules are as follows:

$$x_{i,k}^\tau = x_{i,k-1}^\tau - \eta_l (g_i(x_{i,k-1}^\tau) + h_i^\tau), \tag{12}$$

$$h_i^\tau = c^\tau - c_i, \tag{13}$$

where $g_i(x_{i,k-1}^\tau)$ is the stochastic gradient, h_i^τ is the temporal gradient correction term, c^τ is the global correction term at round τ, c_i is the local correction term of client i, k is the current step number of client updates.

Similar to SCAFFOLD (Karimireddy et al. 2020), the update process for c_i and c^t is as follows:

$$\hat{c_i} = \frac{1}{K\eta_l}(x_i^\tau - x_{i,K}^\tau) - (c^\tau - c_i), \tag{14}$$

Algorithm 1: The whole training procedure of FedAC.

```
1: Input: Randomly initialize model parameters $x^0 = x_i^0$;
   Initialize $c_i$, $c^0$, $m^0$, $v^0$ as 0; Hyperparameters $\eta_g$, $\eta_l$,
   $\beta_1, \beta_2, B, \epsilon$.
2: Output: the final global model $x^T$.
3: Server model aggregation:
4: repeat
5:   if server receives local result from client $i$ then
6:     Server stores ($g(x_i^\tau)$,$\Delta c_i$) from client $i$;
7:     $b = b + 1$;
8:     if $b == |B|$ then
9:       Server computes $w_i^t$ as in Eq.(4) and Eq.(5);
10:      Server updates $x^t$ as in Eq.(7) - Eq.(11);
11:      Server computes $c^t$ as in Eq.(17);
12:      Server distributes ($x^t$, $c^t$) to clients in buffer;
13:      $t = t + 1, b = 0$;
14:    end if
15:  end if
16: until convergence
17: Client model update:
18: for each client $i \in S$ in parallel do
19:   if client receives ($x^t$, $c^t$) from server then
20:     Initialize $x_{i,0}^\tau = x^t, c^\tau = c^t$, $h_i^\tau = c^\tau - c_i$;
21:     for $k = 1, \ldots, K$ do
22:       Client updates $x_{i,k}^\tau$ as in Eq.(12);
23:     end for
24:     Client computes $c_i$ as in Eq.(14) - Eq.(16);
25:     Client computes $g(x_i^\tau)$ as in Eq.(6);
26:     Client uploads local result($\Delta c_i, g(x_i^\tau)$) to server;
27:   end if
28: end for
29: Return $x^T$.
```

$$\Delta c_i = \hat{c_i} - c_i, \tag{15}$$

$$c_i = \hat{c_i}, \tag{16}$$

$$c^t = c^{t-1} + \sum\nolimits_{i \in B} w_i^t \Delta c_i. \tag{17}$$

The c_i in h_i^τ is reset for the client i's next update round and Δc_i is sent back to the server and stored in the buffer.

Usefulness of h_i^τ. To mitigate client drift, the client updates are as follows, without considering the communication cost:

$$x_{i,k}^\tau = x_{i,k-1}^\tau - \frac{\eta_l}{N}\sum_{j=1}^{N} g_j(x_{i,k-1}^\tau). \tag{18}$$

Eq.(18) computes unbiased gradients for the client i. However, it requires communication between client i and all the other clients for each update. To address this, we propose an alternative client update scheme as shown below:

$$c_i \approx g_i(x_{i,k-1}^\tau), \tag{19}$$

$$c^\tau \approx \sum\nolimits_{i \in B} w_i^t \Delta c_i. \tag{20}$$

With this approach, the client updates closely approximate the unbiased estimation in the ideal scenario:

$$g_i(x_{i,k-1}^\tau) - c_i + c^\tau \approx \frac{1}{N}\sum_{j=1}^{N} g_j(x_{i,k-1}^\tau). \tag{21}$$

In Eq.(19), the local correction term c_i can approximate the local average gradient towards the local optimal. In Eq.(17), Δc_i from different clients is multiplied by their weight w_i to weaken the impact of model staleness on the average gradients, and the local correction term c_i is initialized as 0 when the client first starts local update. Therefore, c^τ in Eq.(20) can reflect the average gradient of all clients on the global model x^τ.

During client model training, we use the local correction term c_i to weaken the update of the stochastic gradient $g_i(x^\tau_{i,k-1})$ towards local optima and use the global correction term c^τ to give clients a global view for updating the model towards the global optimal. Therefore, h^τ_i can mitigate the client drift towards local optima and correct client stochastic gradient towards global optima.

Our algorithm for client update with fine-grained gradient correction in AFL can perform fine-grained gradient adjustments at each training step on the client, which allows clients to have a global view during model updates, mitigates the impact of client drift caused by data heterogeneity, and improves the client model quality. The overall training algorithm of FedAC is shown in Algorithm 1.

Experiments

Experimental Settings

Datasets. We utilize three representative FL datasets, namely CIFAR-10 (Krizhevsky 2009), EMNIST-L (Cohen et al. 2017) and Shakespeare (Shakespeare 2002), to evaluate the performance of our method. The division of training and test sets is carried out in the same manner as previous studies (McMahan et al. 2017). In the IID setting, the training samples are randomly assigned and equally distributed among clients. In the Non-IID setting, label ratios are determined by the Dirichlet distribution (Yurochkin et al. 2019), where the parameter α controls the degree of data heterogeneity. For the Shakespeare dataset, we utilize the same approach and employ LEAF for data partitioning (McMahan et al. 2017; Acar et al. 2021; Wang et al. 2022b).

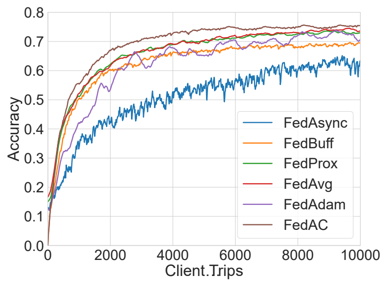
Networks. To thoroughly evaluate the effectiveness of our method, we follow FedAvg (McMahan et al. 2017) and conduct experiments on three network architectures. For the EMNIST-L dataset, we employ a fully-connected network. Specifically, we use a two-layer multi-layer perceptron (MLP) with ReLU activation, which has a total of 92,337 parameters. For the CIFAR-10 dataset, we utilize the same convolutional neural network (CNN), which comprises two 5x5 convolutional layers (with 32 channels for the first layer and 64 for the second, each followed by a 2x2 max pooling), two fully connected layers with ReLU activation, and a final Softmax output layer, with a total of 797,963 parameters. For the Shakespeare dataset, we adopt a stacked character-level LSTM language model, as described in (McMahan et al. 2017).

Baselines. To validate the effectiveness of our method, we compare it against five baseline FL methods, including FedAvg (McMahan et al. 2017), FedAdam (Reddi et al. 2021), FedProx (Li et al. 2020c), FedAsync (Xie, Koyejo, and Gupta 2019), and FedBuff (Nguyen et al. 2022b). FedAvg and FedAdam are classic SFL methods, with FedAdam using momentum to accelerate aggregation. FedProx incorporates a proximal term to constrain the client updates. FedAsync is a well-known AFL framework. FedBuff uses a buffer on the server to aggregate the client update pseudo-gradients. Owing to the differences between SFL and AFL, when comparing these two approaches, we consider "metrics vs the number of client trips" and follow the experimental method as in (Nguyen et al. 2022b). A client trip entails a client receiving the latest model from the server, performing K steps of training on the local dataset, and then uploading the model update back to the server.

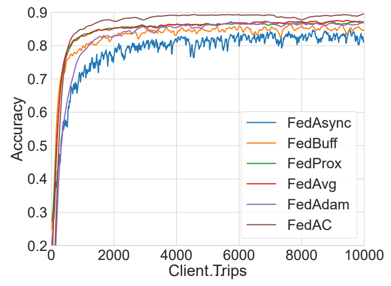
Parameter Settings. To ensure a fair comparison among all methods, the SGD optimizer is used as the client optimizer for all methods. All methods adopt the learning rates $\eta_l = 0.1$ and $\eta_g = 1$. We use the hyperparameters for FedAdam as specified in (Reddi et al. 2021), since FedAdam shows poor training performance under the general hyperparameters mentioned above. The batch size for the experiments is set to 50 and the weight decay is set to 0.002 in local training. For some of the federated settings, the default parameters used are: $N = 100$ for the number of clients, $\alpha = 0.1$ for the Dirichlet parameter, $E = 5$ for the number of client training epochs. The proportion is set to 0.2 for the active clients in SFL. For all experiments, we set the parameters as $\beta_1 = 0.6$, $\beta_2 = 0.9$, $|B| = 20$ and $\epsilon = 10^{-8}$ in FedAC.

Experimental Results Analysis

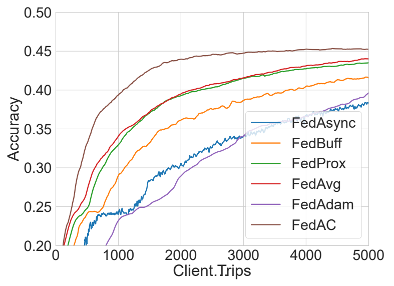
Performance and convergence speed comparison. The convergence curves of FedAC, displayed in Fig. 1, indicate that FedAC surpasses both synchronous and asynchronous baseline methods in terms of convergence speed and accuracy. As clearly shown in Fig. 1a and Fig. 1c, FedAC demonstrates a significant advantage in early convergence speed. By utilizing client weight based on temporal gradient and client pseudo-gradients for calculating prospective momentum, we can predict the next position of the global model at each update. This prediction allows for more effective updates, significantly reducing the number of client trips required for model convergence and enhancing model efficiency. The effectiveness of this approach is demonstrated in Fig. 2, where the baselines need several times more client trips to achieve the specified accuracy compared to our method. Moreover, our proposed client update algorithm corrects stochastic gradients during client updates, guiding clients towards the global optima rather than local optima. This approach greatly alleviates client drift caused by data heterogeneity and enhances local model quality. The use of client weight based on temporal gradient during client updates and server aggregation mitigates the impact of client staleness due to device heterogeneity in AFL. This leads to the learning of higher quality prospective weighted momentum for global model updates and global correction terms for client correction, thereby improving both the convergence speed and accuracy of the model. Additionally, the synergy between the higher quality global model and unbiased local models enhances the calculation of client weight, creating a mutually reinforcing process. This synergy is a key reason



(a) CIFAR-10

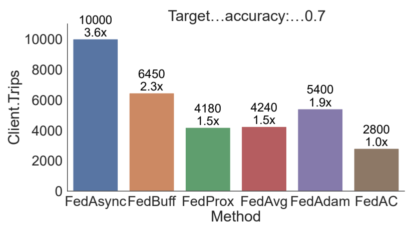


(b) EMNIST-L

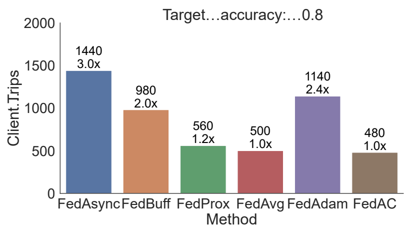


(c) Shakespeare

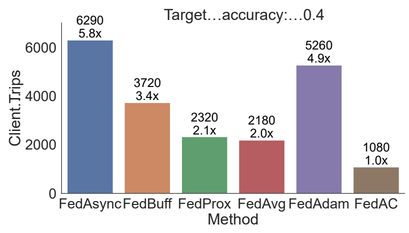
Figure 1: Accuracy curves of FedAC and other baselines on different datasets.



(a) CIFAR-10



(b) EMNIST-L



(c) Shakespeare

Figure 2: Number of client trips required to reach the target accuracy for FedAC and other baselines on different datasets.

Client Amount	Accuracy (%)	client trips (multiplier)					
		FedAC	FedAvg	FedAdam	FedProx	FedAsync	FedBuff
50	0.65	**900 (1×)**	1240 (1.38×)	1540 (1.71×)	1260 (1.40×)	2012 (2.24×)	1470 (1.63×)
	0.6	**650 (1×)**	800 (1.23×)	1300 (2.00×)	800 (1.23×)	1187 (1.83×)	790 (1.21×)
100	0.65	**1720 (1×)**	2240 (1.30×)	2260 (1.31×)	2120 (1.23×)	5000+ (2.90×)	3050 (1.77×)
	0.6	**1320 (1×)**	1500 (1.14×)	1840 (1.39×)	1380 (1.05×)	4210 (3.19×)	2150 (1.63×)
200	0.55	**1280 (1×)**	1660 (1.30×)	2600 (2.03×)	1740 (1.36×)	5000+ (3.90×)	3100 (2.42×)
	0.5	**900 (1×)**	1120 (1.24×)	1600 (1.78×)	1060 (1.18×)	5000+ (5.56×)	2250 (2.50×)
500	0.5	**1800 (1×)**	2200 (1.22×)	4620 (2.57×)	2320 (1.29×)	5000+ (2.78×)	5000+ (2.78×)
	0.35	**800 (1×)**	1060 (1.33×)	1640 (2.05×)	1040 (1.3×)	5000+ (6.25×)	2860 (3.58×)

Table 1: Number of client trips to reach target accuracy for FedAC and other baselines in federated settings with different numbers of clients on CIFAR-10. The values in parentheses represent the multiples of client trips compared to our method when reaching the target accuracy.

why FedAC significantly outperforms baselines in terms of convergence speed and accuracy.

Scalability analysis. Table 1 describes the client trips required to achieve the target accuracy for FedAC and baselines. Notably, regardless of the number of clients involved, baseline methods consistently demand significantly more client trips to achieve the target accuracy in contrast to FedAC. In practical FL scenarios, a vast number of clients often participate. By examining the performance of global model training under varying client counts, we can aptly assess the scalability and real-world applicability of the FL framework. In AFL, a high number of clients exacerbates the staleness of local models, and training on clients with small and Non-IID datasets also exacerbates client drift, which makes the global model updates easily affected by local models and reduces the learning quality of the global model. This can be seen from the results of FedAsync in Table 1. Compared with SFL methods, our method FedAC emerges as a more adaptable and efficient choice for real-world FL environments, outclassing SFL-based strategies such as FedAvg, FedAdam, and FedProx. This superior performance indicates that FedAC effectively handles the challenges posed by data heterogeneity and model staleness. Moreover, it maintains its scalability irrespective of the number of clients participating in FL.

Robustness on data heterogeneity. Table 2 shows the accuracy of FedAC and baselines under different levels of data heterogeneity when the number of client trips is 3000. It can be observed that FedAC achieves the highest accuracy. Compared to baselines, FedAC achieves the smallest difference

Method	α			
	α=0.1	α=1	α=10	IID
FedAvg	66.8	76.1	76.9	77.8
FedADAM	62.9	73.8	78.8	79.1
FedProx	65.7	75.7	76.6	77.6
FedAsync	53.3	65.5	69.4	70.5
FedBuff	63.1	73.2	75.8	76.2
FedAC	**70.6**	**79.6**	**80.1**	**80.3**

Table 2: Accuracy at 3000 client trips for all the methods with different levels of data heterogeneity α.

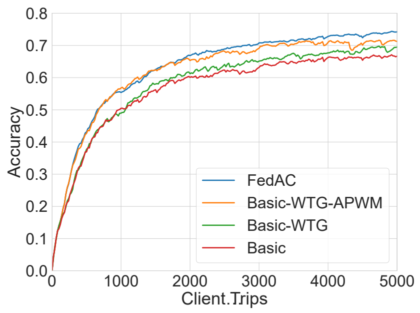


Figure 3: Ablation study results on CIFAR-10.

in accuracy between the non-IID setting such as $\alpha = 0.1$ and the IID setting. This demonstrates that our client update with fine-grained gradient correction algorithm effectively alleviates the client drift caused by data heterogeneity through the use of global and local correction terms. Comparing the performance of different methods in the IID setting in Table 2, we can draw the following conclusions: in contrast to SFL algorithms, which do not suffer from client drift when the client data distributions are consistent, FedAC achieves higher accuracy values at the specified number of communication rounds, demonstrating its ability to address the problem of local model staleness and the effectiveness of the client weights evaluation. In summary, FedAC demonstrates robustness to different levels of data heterogeneity and can improve the efficiency of AFL.

Ablation Study. We conduct an ablation study to demonstrate the effectiveness of each module in our method and design several degraded methods for comparison. The "Basic" model in Fig. 3 is FedBuff, which serves as the foundation for our method. Basic-WTG represents the basic framework combined with "Client Weight Evaluation Based on Temporal Gradient". Basic-WTG-APWM represents the combination of "Adaptive Server Update with Prospective Weighted Momentu" on the basis of Basic-WTG. FedAC represents our complete FL framework. The accuracy curves of each module of FedAC are shown in Fig. 3. By comparing the results of the ablation experiments, we observe that adding the "Client Weight Evaluation Based on Temporal Gradient" module (Basic-WTG) to the Basic model can better evaluate client staleness, thus improving model training quality and convergence speed. Furthermore, adding the "Adaptive Server Update with Prospective Weighted Momentum" module (Basic-WTG-APWM) can accelerate the global model update, but the accuracy is slightly lower than that of FedAC due to the lack of client offset correction. Ultimately, when all modules are activated, FedAC attains the quickest convergence and the pinnacle of accuracy.

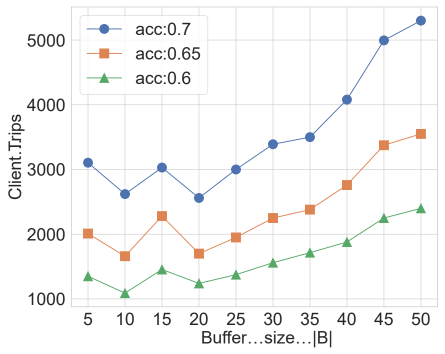


Figure 4: Impact of buffer size $|B|$ on client trips to reach target accuracy on CIFAR-10.

Analysis of buffer size. Fig. 4 presents the impact of the buffer size $|B|$ on the efficiency of AFL by analyzing the number of client trips required to reach the target accuracy. Notably, when the buffer size $|B|$ is calibrated to either 10 or 20, FedAC surpasses other buffer size configurations in minimizing the client trips required to attain the three target accuracies. More specifically, targeting an accuracy of 0.7, the configuration with $|B| = 20$ necessitates fewer client trips than any other buffer size. In light of these findings, to achieve better efficacy and performance from FedAC, we choose a buffer size of $|B| = 20$ throughout the experiment.

Conclusion

In this paper, we propose efficient asynchronous federated learning with prospective momentum aggregation and fine-grained correction to address the efficiency and performance issues in AFL caused by data heterogeneity and device heterogeneity. We use client weight evaluation based on temporal gradient to calculate prospective weighted momentum for adaptive server update to improve the convergence speed and AFL efficiency, and fine-grained gradient correction during client update to mitigate client drift and enhance the AFL performance. Extensive experiments on both real and synthetic datasets demonstrate that FedAC significantly improves the efficiency and performance of asynchronous federated learning.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62272058, 62192784, U22B2038, 62172056), the 8th Young Elite Scientists Sponsorship Program by CAST (2022QNRC001), and in part by the NSERC Discovery Grant program.

References

Acar, D. A. E.; Zhao, Y.; Matas, R.; Mattina, M.; Whatmough, P.; and Saligrama, V. 2021. Federated Learning Based on Dynamic Regularization. In *International Conference on Learning Representations*.

Chai, Z.; Chen, Y.; Zhao, L.; Cheng, Y.; and Rangwala, H. 2020. Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data. *ArXivorg*.

Chen, Y.; Ning, Y.; Slawski, M.; and Rangwala, H. 2020. Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*, 15–24. IEEE.

Chen, Y.; Sun, X.; and Jin, Y. 2019. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. *IEEE transactions on neural networks and learning systems*, 31(10): 4229–4238.

Cohen, G.; Afshar, S.; Tapson, J.; and Van Schaik, A. 2017. EMNIST: Extending MNIST to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, 2921–2926. IEEE.

Dozat, T. 2016. Incorporating Nesterov Momentum into Adam. In *Proceedings of the 4th International Conference on Learning Representations*, 1–4.

Gao, L.; Fu, H.; Li, L.; Chen, Y.; Xu, M.; and Xu, C.-Z. 2022. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10112–10121.

Guan, Z.; Li, Y.; Xue, Z.; Liu, Y.; Gao, H.; and Shao, Y. 2021. Federated Graph Neural Network for Cross-graph Node Classification. In *2021 IEEE 7th International Conference on Cloud Computing and Intelligent Systems (CCIS)*, 418–422.

He, J.; Wang, T.; Min, Y.; and Gu, Q. 2022. A Simple and Provably Efficient Algorithm for Asynchronous Federated Contextual Linear Bandits. *arXiv preprint arXiv:2207.03106*.

Hsu, T.-M. H.; Qi, H.; and Brown, M. 2019. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*.

Jiang, Z.; Wang, W.; Li, B.; and Li, B. 2022. Pisces: Efficient federated learning via guided asynchronous training. In *Proceedings of the 13th Symposium on Cloud Computing*, 370–385.

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

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, 5132–5143. PMLR.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koloskova, A.; Stich, S. U.; and Jaggi, M. 2022. Sharper convergence guarantees for asynchronous sgd for distributed and federated learning. *Advances in Neural Information Processing Systems*, 35: 17202–17215.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. 32–33.

Lee, J.-w.; Oh, J.; Shin, Y.; Lee, J.-G.; and Yoon, S.-Y. 2020. Accurate and fast federated learning via iid and communication-aware grouping. *arXiv preprint arXiv:2012.04857*.

Li, C.; and Wang, H. 2022. Asynchronous upper confidence bound algorithms for federated linear bandits. In *International Conference on Artificial Intelligence and Statistics*, 6529–6553. PMLR.

Li, Q.; Zhu, W.; Wu, C.; Pan, X.; Yang, F.; Zhou, Y.; and Zhang, Y. 2020a. InvisibleFL: federated learning over non-informative intermediate updates against multimedia privacy leakages. In *Proceedings of the 28th ACM International Conference on Multimedia*, 753–762.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020b. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine*, 37(3): 50–60.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020c. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2: 429–450.

Li, Y.; Li, W.; and Xue, Z. 2022. Federated Learning with Stochastic Quantization. *Int. J. Intell. Syst.*, 37(12): 11600–11621.

Long, Y.; Xue, Z.; Chu, L.; Zhang, T.; Wu, J.; Zang, Y.; and Du, J. 2023. FedCD: A Classifier Debiased Federated Learning Framework for Non-IID Data. In *Proceedings of the 31st ACM International Conference on Multimedia*, MM '23, 8994–9002. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701085.

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

Nesterov, Y. E. 1983. A method for solving the convex programming problem with convergence rate O (1/k^ 2). In *Dokl. akad. nauk Sssr*, volume 269, 543–547.

Nguyen, J.; Malik, K.; Zhan, H.; Yousefpour, A.; Rabbat, M.; Malek, M.; and Huba, D. 2022a. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*, 3581–3607. PMLR.

Nguyen, J.; Malik, K.; Zhan, H.; Yousefpour, A.; Rabbat, M.; Malek, M.; and Huba, D. 2022b. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*, 3581–3607. PMLR.

Reddi, S. J.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečný, J.; Kumar, S.; and McMahan, H. B. 2021. Adaptive Federated Optimization. In *International Conference on Learning Representations*.

Sattler, F.; Müller, K.-R.; and Samek, W. 2020. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 32(8): 3710–3722.

Shakespeare, W. 2002. *The Complete Pelican Shakespeare*. Penguin.

Shi, G.; Li, L.; Wang, J.; Chen, W.; Ye, K.; and Xu, C. 2020. HySync: Hybrid federated learning with effective synchronization. In *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 628–633. IEEE.

So, J.; Ali, R. E.; Güler, B.; and Avestimehr, A. S. 2021. Secure aggregation for buffered asynchronous federated learning. *arXiv preprint arXiv:2110.02177*.

Wang, H.; Li, R.; Li, C.; Zhou, P.; Li, Y.; Xu, W.; and Guo, S. 2022a. Gradient Scheduling With Global Momentum for Asynchronous Federated Learning in Edge Environment. *IEEE Internet of Things Journal*, 9(19): 18817–18828.

Wang, Q.; Yang, Q.; He, S.; Shui, Z.; and Chen, J. 2022b. AsyncFedED: Asynchronous Federated Learning with Euclidean Distance based Adaptive Weight Aggregation. *arXiv preprint arXiv:2205.13797*.

Wu, W.; He, L.; Lin, W.; Mao, R.; Maple, C.; and Jarvis, S. 2020. SAFA: A semi-asynchronous protocol for fast federated learning with low overhead. *IEEE Transactions on Computers*, 70(5): 655–668.

Xie, C.; Koyejo, S.; and Gupta, I. 2019. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*.

Xu, C.; Qu, Y.; Xiang, Y.; and Gao, L. 2021. Asynchronous federated learning on heterogeneous devices: A survey. *arXiv preprint arXiv:2109.04269*.

Yurochkin, M.; Agarwal, M.; Ghosh, S.; Greenewald, K.; Hoang, N.; and Khazaeni, Y. 2019. Bayesian nonparametric federated learning of neural networks. In *International Conference on Machine Learning*, 7252–7261. PMLR.

Zang, Y.; Xue, Z.; Ou, S.; Long, Y.; Zhou, H.; and Du, J. 2023. FedPcf: An Integrated Federated Learning Framework with Multi-Level Prospective Correction Factor. In *Proceedings of the 2023 ACM International Conference on Multimedia Retrieval*, ICMR '23, 490–498. New York, NY, USA: Association for Computing Machinery. ISBN 9798400701788.

Zhang, Q.; Gu, B.; Deng, C.; and Huang, H. 2021. Secure bilevel asynchronous vertical federated learning with backward updating. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 10896–10904.

Zhou, C.; Tian, H.; Zhang, H.; Zhang, J.; Dong, M.; and Jia, J. 2021. TEA-fed: time-efficient asynchronous federated learning for edge computing. In *Proceedings of the 18th ACM International Conference on Computing Frontiers*, 30–37.

Zhuang, W.; Wen, Y.; Zhang, X.; Gan, X.; Yin, D.; Zhou, D.; Zhang, S.; and Yi, S. 2020. Performance optimization of federated person re-identification via benchmark analysis. In *Proceedings of the 28th ACM International Conference on Multimedia*, 955–963.

Zhuo, Y.; and Li, B. 2021. Fedns: Improving federated learning for collaborative image classification on mobile clients. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, 1–6. IEEE.