electronics

MDPI

Article

Fed2A: Federated Learning Mechanism in Asynchronous and Adaptive Modes

Sheng Liu, Qiyang Chen and Linlin You *

School of Intelligent Systems Engineering, Sun Yat-Sen University, Guangzhou 510006, China; liush235@mail2.sysu.edu.cn (S.L.); chenqy87@mail2.sysu.edu.cn (Q.C.)
* Correspondence: youllin@mail.sysu.edu.cn

Abstract: Driven by emerging technologies such as edge computing and Internet of Things (IoT), recent years have witnessed the increasing growth of data processing in a distributed way. Federated Learning (FL), a novel decentralized learning paradigm that can unify massive devices to train a global model without compromising privacy, is drawing much attention from both academics and industries. However, the performance dropping of FL running in a heterogeneous and asynchronous environment hinders its wide applications, such as in autonomous driving and assistive healthcare. Motivated by this, we propose a novel mechanism, called Fed2A: Federated learning mechanism in Asynchronous and Adaptive Modes. Fed2A supports FL by (1) allowing clients and the collaborator to work separately and asynchronously, (2) uploading shallow and deep layers of deep neural networks (DNNs) adaptively, and (3) aggregating local parameters by weighing on the freshness of information and representational consistency of model layers jointly. Moreover, the effectiveness and efficiency of Fed2A are also analyzed based on three standard datasets, i.e., FMNIST, CIFAR-10, and GermanTS. Compared with the best performance among three baselines, i.e., FedAvg, FedProx, and FedAsync, Fed2A can reduce the communication cost by over 77%, as well as improve model accuracy and learning speed by over 19% and 76%, respectively.

Keywords: federated learning; asynchronous federated learning; adaptive uploading; adaptive aggregation

Citation: Liu, S.; Chen, Q.; You, L. Fed2A: Federated Learning Mechanism in Asynchronous and Adaptive Modes. *Electronics* **2022**, *11*, 1393. https://doi.org/10.3390/electronics11091393

Academic Editor: Rui Pedro Lopes

Received: 26 March 2022
Accepted: 25 April 2022
Published: 27 April 2022

1. Introduction

The number of autonomous vehicles, IoT (Internet of Things) devices, mobile phones, and various sensors has increased significantly because of the fast development of Artificial Intelligence, ICT (Information and Communication Technology), electronics, and other advanced technologies. Along with this trend, massive, distributed, and valuable data are generated and become sensitive to be collected and processed at the data center. In this context, a decentralized approach, called Federated Learning (FL), is proposed to train a global model by collaborating with various clients in a privacy-preserving way [1]. In FL, learning participants can train and upload local models by utilizing their local resources (i.e., data, computation, and communication capabilities). Meanwhile, the collaborator can receive and aggregate local models to build a global model iteratively. Hence, sensitive data are protected without sharing them explicitly, and distributed computing powers are jointly utilized to ease the workload of central servers. As a key enabler of intelligent services, FL has already been applied in various domains [2], e.g., (1) in transportation to enable autonomous vehicles to behave in a smarter way while protecting user trajectories and other sensitive information [3]; (2) in healthcare to train disease prediction models while securing patient data [4]; (3) in manufacturing to detect intrusions or anomalies while utilizing sensed data of edge devices [5].

However, existing research and applications of FL focus more on the synchronous mode. It ensures all clients work at the same pace. Such that, stragglers may impede

the whole training process. On the contrary, in the asynchronous mode, clients can work individually, and the collaborator can start the model aggregation once the pre-defined condition is met without waiting [6,7]. For example, to train an image classification model by collaborating massive mobile phones under the FL framework, since these devices vary in their local resources, such as local data sizes, computing capabilities, battery levels, and network situations (4G, 5G, WiFi, etc.), there may exist some incompetent clients, i.e., the stragglers, requiring much longer learning time, or failing to complete the task. In synchronous FL, the stragglers become the performance bottleneck of FL to support such a learning task. Whereas, in asynchronous FL (AFL), the influence of stragglers can be remedied, as each client can train and upload the local model separately without any suspension. Such that, AFL is more suitable to support services consisting of devices with heterogeneous capabilities.

Due to such advantages, AFL is now in the spotlight, but it encounters two critical challenges [8–10], namely:

- *How to reduce the communication cost without damaging the overall learning performance?* Since the communication capacities of AFL clients are limited, unstable, and changeable over time, and an incaution reduction causing the loss of vital information may impact the overall learning performance, it becomes critical to optimize the usage of communication resources of learning participants rationally.
- *How to improve the model performance by harnessing temporal and informative attributes of local parameters?* Since local parameters are subject to Non-iid (Non independent and identically distributed) data, and their created and received time may vary, it becomes essential to address these variances during the model aggregation process comprehensively.

To tackle the challenges, several strategies have been proposed, e.g., a layer-wise model update strategy and a temporally weighted aggregation strategy are designed to train DNNs (Deep Neural Networks) [11]. Even though these solutions can reduce communication costs, side effects are observed that the model accuracy is decreased and its growth curve becomes vibrated. Two limitations are responsible for these solutions, namely, (1) the proper uploading frequencies of different layers have not been studied in-depth, hence the missing of some valuable information deteriorates the model accuracy, and (2) the aggregation procedure utilizes only temporal or informative attributes without measuring them jointly, hence, related biases unsteady the learning process.

In this context, a novel mechanism is required to reduce communication costs without compromising learning performance by optimizing the layer uploading strategy, and in the meantime, improve learning performance by harnessing both the temporal and informative heterogeneity of local parameters. To fill this gap, this paper proposes Fed2A: Federated learning mechanism in Asynchronous and Adaptive modes. Specifically, first, it designs and implements a two-stage asynchronous learning procedure, including:

- ***Local uploading stage***. AFL clients can work to train local models and upload them to the AFL collaborator concurrently and individually;
- ***Global aggregation stage***. The AFL collaborator can receive local parameters from AFL clients continuously while aggregating received parameters simultaneously.

Moreover, it proposes and integrates three adaptive strategies to support the local uploading and global aggregation stages to train DNNs, namely:

- ***PLU****: Periodic layer uploading strategy*. It divides the total global rounds into multiple periods, and parameters of deep layers will only be uploaded in specific rounds of a period to save communication costs.
- ***TVW****: Time variety weighting strategy*. It creates temporal weights reflecting the difference between the created and received times of local parameters to assist the model aggregation.
- ***RCE****: Representational consistency enhancing strategy*. It measures the informative importance for each received layer of DNNs in a global round and uses it to steer the aggregation direction.

Accordingly, the main contributions of this paper are summarized as follows:

- We design a novel AFL mechanism, called Fed2A, to support distributed learning in heterogeneous environments with communication cost reduced and model performance improved, as Fed2A can support the local uploading and global aggregation stages asynchronously and adaptively.
- We optimize the local uploading stage by splitting the uploading of deep and shallow layers and reducing the uploading frequency of deep layers appropriately.
- We enhance the global aggregation stage by designing delay functions to punish stale models, and utilizing representational consistency between local and global layers to guide the learning direction.
- We conduct holistic experiments to evaluate Fed2A based on three standard datasets (i.e., FMNIST, CIFAR-10, and GermanTS). Compared to three state-of-the-art baselines (i.e., FedAvg, FedProx, and FedAsync), Fed2A can (1) reduce the communication cost by about 77.30%, (2) improve the model accuracy by about 19.20%, and (3) boost the training speed by about 76.62% simultaneously and respectively.

The remainder of this paper is organized as follows. First, Section 2 summarizes related work about communication cost reduction and model aggregation optimization. Second, Fed2A is presented and evaluated in Sections 3 and 4, respectively. Finally, Section 5 concludes the work and sketches the future research directions.

2. Related Work

To improve the overall performance of AFL, communication cost reduction strategies in the local uploading stage, and aggregation weights optimization strategies addressing temporal or informative heterogeneity issues in the global aggregation stage are studied.

2.1. Communication Cost Reduction

DNN tends to have massive parameters to be trained and transmitted in AFL. In general, three methods, i.e., model compression, frequency reduction, and training split, are widely discussed to reduce the consumption of limited communication resources of clients.

First, for model compression, Caldas et al. [12] introduced lossy compression and federated dropout strategies to decrease the number of uploaded parameters; Wang et al. [13] utilized singular value decomposition to discover and transfer only useful gradients; Sattler et al. [14] and Asad et al. [15] designed sparse compression schemes to reduce redundant parameters; Lu et al. [16] proposed threshold-based compression strategies to only transmit the qualified gradients.

Second, regarding frequency reduction, some scholars increased the local epochs for fast model convergence, in turn, to reduce the communication frequency. However, these kinds of solutions consume more computing powers [17]. To avoid that, several uploading frequency reduction strategies were studied by utilizing cross-entropy loss [18] or convergence speed feedback [19,20].

Finally, as for the last method, each client can train and upload a subset of the model (i.e., the shallow and deep layers can be split) to save related costs. Kang et al. [21], for example, distributed a model according to data sizes of clients, and Wang et al. [22] divided the global model into branches according to sample categories.

2.2. Aggregation Weights Optimization

Since the model parameters from clients are uploaded asynchronously with various latencies, several temporally weighted strategies were proposed to tackle such an issue. For example, Xie et al. [7] designed an optimization algorithm, called FedAsync, which can update the global model right after receiving an arbitrary local model, and uses a mixing hyperparameter related to the training start and received time of the local model to balance the weights applied to the current global model and received local model during the aggregation; Chen et al. [11] also proposed a temporally weighted aggregation strategy based on the assumption that newer parameters deserve larger weights.

Moreover, since data quality varies among distributed AFL clients, several informative weights are designed to boost model performance. For example, Wang et al. [23] proposed a reputation-based strategy to assign weights by estimating the contributions of clients; Chen et al. [24] presented a quantization-driven algorithm to allocate weights by using the instantaneous quantization errors reported by clients; Wang et al. [25] designed an attention-based strategy to optimize aggregation weights by avoiding the imbalance in local models; Li et al. [26] introduced an empirical risk-based scheme to adjust weights automatically by evaluating reliability and corruption levels of local data.

In summary, as summarized in Table 1, model compression, frequency reduction, and training split are widely used to reduce the communication consumption that occurs during the interaction between learning participants and the collaborator. Second, several temporally and informatively weighted strategies are also studied to optimize the aggregation process for a performance boost. However, these strategies are generally with side effects such as compromised model performance, and lack of comprehensiveness. Hence, a novel mechanism that can not only reduce communication costs significantly but also improve learning performance thoroughly needs to be studied, which is the main purpose of this paper.

Table 1. The overview of reviewed works (◯, not supported).

Related Work	Communication Cost Reduction	Aggregation Based on Temporal Attributes	Aggregation Based on Informative Attributes
Caldas et al. [12]	Dropout compression	◯	◯
Wang et al. [13]	SVD compression	◯	◯
Sattler et al. [14]	Sparse compression	◯	◯
Asad et al. [15]	Sparce compression	◯	◯
Lu et al. [16]	Threshold compression	◯	◯
Zhu et al. [17]	Frequency reduction	◯	◯
Huang et al. [18]	Frequency reduction	◯	◯
Wang et al. [19]	Frequency reduction	◯	◯
Wang et al. [20]	Frequency reduction	◯	◯
Kang et al. [21]	Training split	◯	◯
Wang et al. [22]	Training split	◯	◯
Xie et al. [7]	◯	Exp/polynomial/ inverse/hinge delay function	◯
Chen et al. [11]	Frequency reduction and training split	Exp delay function	◯
Wang et al. [23]	◯	◯	Reputation-based
Chen et al. [24]	◯	◯	Quantization-driven
Wang et al. [25]	◯	◯	Attention-based
Li et al. [26]	◯	◯	Empirical risk-based
Fed2A (Proposed)	Frequency reduction and training split	Exp/inverse/log delay function	Representational consistency-based

3. Introduction of Fed2A

This section first describes the problem formulation and then introduces the asynchronous learning procedure implemented in Fed2A. Moreover, three adaptive strategies of Fed2A, i.e., PLU, TVW, and RCE, are presented. Finally, Fed2A, with the three strategies utilized together, is discussed. For the sake of readability, the notations used in this section are summarized in Table 2.

Table 2. The notations used in Fed2A.

Notation	Description
D_k	The local dataset of client k
R	The total global rounds
P	The total rounds in a PLU period
D	The last D rounds in a PLU period to upload deep layers
s	The parameter size of shallow layers
d	The parameter size of deep layers
ω_{t+1}	The global model of the $(t+1)$th global round
ω_k	The local model of client k
ω_k^l	The layer l of client k
K	The total local models received in a global round
t_k	The generated timestamp of client k
n_k	The data size of client k
n_t	The total data size of the tth global round
S	The stimulus set
m	The total stimuli
L	The total number of layers of trained DNN
$v_l(s_i)$	The output of layer l on stimulus i
O_l	The output dimension of layer l
$rc_{k,t}^l$	The representational consistency of layer l for client k in round t

3.1. Problem Formulation

We consider applying AFL in heterogeneous environments to train DNNs. Assume that there are total K_0 clients and one collaborator, and each client k trains a DNN with total L layers on its local dataset D_k with the size of $n_k = |D_k|$. After client k receives a global model w_{t_k} from the collaborator in global round t_k, it will train the local model w_k by using a predefined optimization function f, such as SGD (stochastic gradient descent), as shown by Formula (1).

$$w_k \leftarrow f(w_{t_k}, D_k) \tag{1}$$

After the local training, w_k will be uploaded to the collaborator. Once an aggregation trigger is satisfied, such as K, updates are received in the tth round, the collaborator will update the global model according to (2), where α_k is the aggregation weight of client k.

$$w_{t+1} = \sum_{k=1}^{K} (\alpha_k \times w_k) \tag{2}$$

In general, once the w_k is trained and uploaded, the core challenge is finding how to optimize the coefficient α_k adaptively in each learning round to update the new global model efficiently and effectively. We identify the following two main factors that affect α_k:

- ***Temporal heterogeneity***. It stands for the difference between the created time of w_k at the client-side (t_k) and the received time at the collaborator-side (t) to measure the staleness of the local model.
- ***Informative heterogeneity***. It stands for the difference between local model w_k and the latest global model w_t, and intuitively, such a divergence needs to be measured per layer.

Hence, we rewrite Formula (2) into Formula (3), where $g(\cdot)$ is the function to compute the aggregation weight for each layer of client k in a round by addressing both temporal and informative heterogeneities.

$$w_{t+1} = \sum_{l=1}^{L} \sum_{k=1}^{K} (\alpha_k^l \times w_k^l), \quad \alpha_k^l \leftarrow g(t_k, t, w_t^l, w_k^l) \tag{3}$$

Therefore, this paper investigates how to upload and aggregate local layers adaptively with heterogeneous attributes for AFL.

3.2. Two-Stage Asynchronous Learning Procedure of Fed2A

As illustrated in Figure 1, it contains two major stages, namely:

- *Local uploading stage on the client side*. The AFL clients can work individually with more flexibility, which means they can start the training and uploading of local models based on their own decisions, and the synchronous step defined in conventional FL methods is no anymore necessary;
- *Global aggregation stage on the collaborator side*. The AFL collaborator can continuously receive parameters from AFL clients. Moreover, it also can start the aggregation once a predefined condition is satisfied, e.g., a certain number of updates are received or the maximum waiting time is reached. Finally, a new global round starts when the updated global model is broadcasted to the clients, who have made contributions.

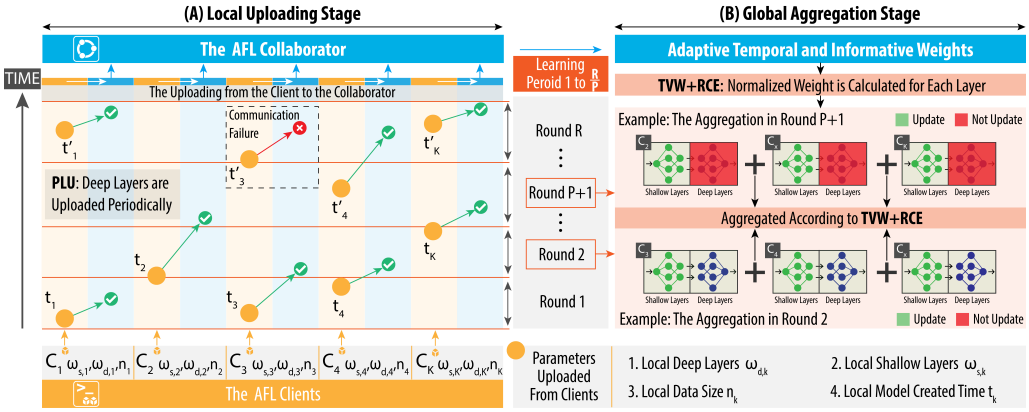

Figure 1. The two-stage asynchronous learning procedure and the allocation of the three adaptive strategies of Fed2A. (**A**) Local Uploading Stage to train and upload local models according to PLU, and (**B**) Global Aggregation Stage to aggregate received local models according to TVW and RCE.

In this context, Fed2A can (1) alleviate the adverse effects of stragglers compared with the synchronous mode, since the aggregation is unblocked; and (2) maintain a stable learning process compared with canonical asynchronous modes, such as FedAsync [7], since more local models can be aggregated in a global round for a more significant update.

Moreover, to reduce communication cost and improve model performance simultaneously under such an asynchronous mode, Fed2A also consists of three adaptive strategies for local uploading and global aggregation, which are discussed in the following subsection.

3.3. Adaptive Uploading and Aggregation Strategies of Fed2A

There are three adaptive strategies, namely, PLU for local updating, and TVM and RCE for global aggregation.

3.3.1. PLU: Periodic Layer Uploading Strategy

It is designed based on the observation that the optimization of the update frequency of shadow and deep layers of DNN can reduce the overall communication cost of FL without damaging the model performance [11]. On the one hand, shallow layers (such as convolutional layers) learn more general features (more critical to the global model) from the training data, while having fewer learning parameters. On the other hand, deep layers (such as fully connected layers) extract ad hoc features closely related to specific data distribution, hence having many more parameters to train. In this context, as shown in Figure 1, the PLU, denoted as $PLU(R, P, D)$, divides total R global rounds into $\frac{R}{P}$ learning periods, and in each learning period, shallow layers will be transmitted in the all P rounds

to learn general features consistently, while deep layers will only be uploaded in the last D rounds to update ad hoc features periodically. Note that since deep layers need to be initialized at the beginning, PLU allows all layers to be uploaded to learn the global model in the first learning period.

Moreover, assume that shallow and deep layers have s and d parameters respectively, accordingly, the total communication cost by using PLU is $(R \times s + (P - D) \times d + \frac{R}{P} \times D \times d)$. Compared to the conventional scenario, whose cost is $(R \times (s + d))$, the cost reduction is $(\frac{R \times D \times d}{P} - (P - D) \times d)$.

Finally, since d is the major component of communication cost, the cost reduction can be considerable when D is smaller than P. However, if the uploading frequency of deep layers is too low, the necessary information will be lost to impact the performance of the global model, which is not ideal. Hence, PLU should select a proper set of P and D to achieve an equilibrium, where the saving on communication cost and the improvement on model performance are both maximized from the system point of view. The analysis of PLU can be found in Section 4.2.

3.3.2. TVM: Time Variety Weighted Strategy

As shown in Figure 2, since each AFL client can join the learning process freely, the collaborator, which aggregates local parameters for the global model, may receive local parameters with various timestamps. To mediate the temporal variances under such circumstances, a time variety weighted (TVW) strategy is proposed to process local parameters of AFL clients with a temporal weight created according to the assumption that the latest parameter shall have the highest weight among received parameters as it contains the latest information.

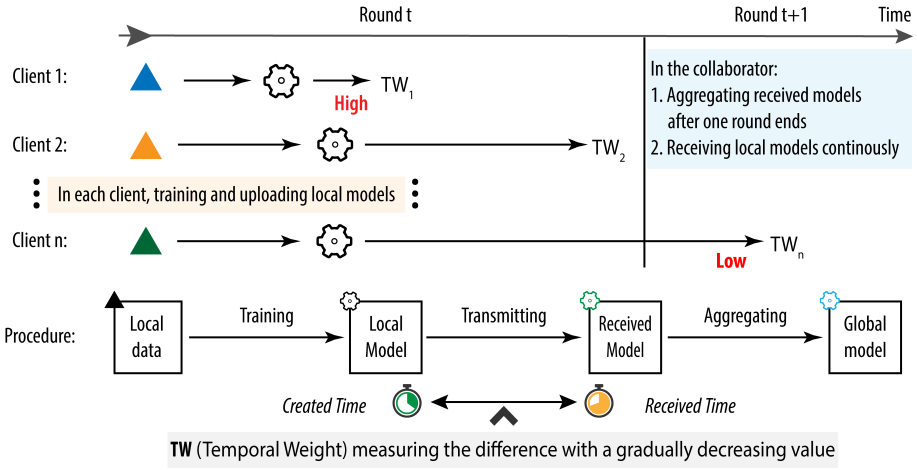

Figure 2. The schematic diagram of TVW.

Accordingly, TVW is defined by Formula (4), where K denotes the number of clients with their parameters received by the collaborator in the tth global round; n_k is the data size of the kth client; n_t is the data size of all clients participating in the tth global round; ω_k denotes the local model received from the kth client; ω_{t+1} denotes the aggregated global model after the tth round; $\overline{TW_{k,t}}$ is the normalized temporal weight of the kth client in the tth global round calculated by the AFL collaborator; $TW_{k,t}$ is the original weight, whose sum is TW_t; e stands for the natural logarithm; and t_k denotes the global round, in which the parameter of the kth client is generated. Note that three typical value decreasing functions

$f(t,k)$, i.e., exponential function, inverse function, and logarithmic function, can be used to calculate the weights.

$$\omega_{t+1} \leftarrow \sum_{k=1}^{K} (\overline{TW_{k,t}} \times \omega_k)$$

$$\begin{cases} \overline{TW_{k,t}} = \frac{TW_{k,t}}{TW_t} \\ TW_{k,t} = \frac{n_k}{n_t} \times \frac{f(t,k)}{f(t)} \\ TW_t = \sum_{k=1}^{K} TW_{k,t} \\ f(t,k) = \begin{cases} (\frac{e}{2})^{-(t-t_k)}, & \exp \\ \frac{1}{t-t_k+1}, & \text{inv} \\ \frac{1}{log(t-t_k+1)+1}, & \log \end{cases} \\ f(t) = \sum_{k=1}^{K} f(t,k) \end{cases} \tag{4}$$

In summary, TVW aggregates model parameters of clients received in an asynchronous global round by manipulating a normalized weight, whose value decreases according to the temporal difference between the timestamps of two rounds, when the local parameters are learned and received. As for the performance of TVW, it is analyzed in Section 4.3.

3.3.3. RCE: Representational Consistency Enhancing Strategy

In general, each layer of local models may make distinctive contributions to the global model, e.g., the difference between shallow and deep layers of DNN [11]. Since (1) layers of the global model are updated by aggregating corresponding layers of local models uploaded from AFL clients, and (2) the importance of layers in local models may vary from each other, instead of aggregating them averagely, a Representational Consistency Enhancing (RCE) strategy is designed to highlight essential layers of local models by comparing the differences with their corresponding layers of current global model in use.

As shown in Figure 3, such a difference can be measured based on multivariate analysis techniques, such as representational similarity analysis [27], which can compare the representational consistency [28] of layers. Specifically, a representational dissimilarity matrix (RDM) can be computed to characterize the internal stimulus representations of layers based on pairwise response differences in a learning round. Note that if two stimuli are represented similarly in a given layer, then the distance between their outputs will be small.

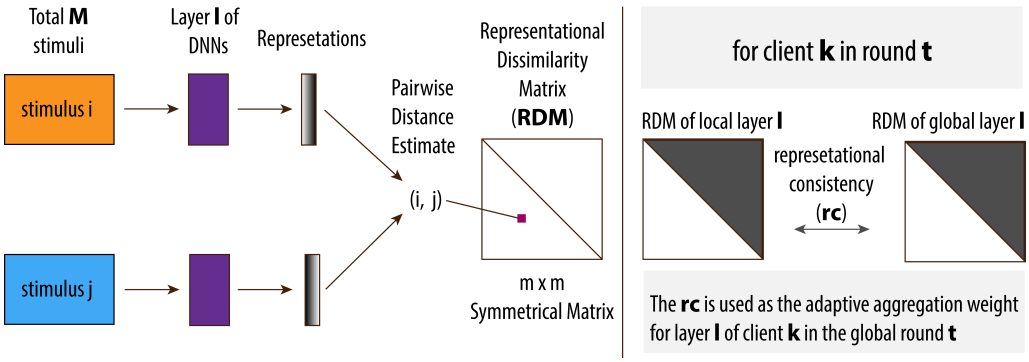

Figure 3. The schematic diagram of RCE.

First, for a given set of stimuli $S = \{s_1, s_2, \ldots, s_m\}$, a layer l, denoted as v_l, can be defined as a vector of outputs on S by Formula (5).

$$v_l = (v_l(s_1), v_l(s_2), \ldots, v_l(s_m)) \tag{5}$$

Second, a symmetrical RDM $(m \times m)$ storing $(\frac{m^2}{2} - m)$ unique pairwise distances between two representations of the layer l to the set of stimuli S, e.g., the distance between $v_l(s_i)$ and $v_l(s_j)$, can be calculated by Formula (6). As for the pairwise distance, it can be measured according to the correlation distance (Formula (7)), cosine distance (Formula (8)), or Euclidean distance (Formula (9)).

$$RDM[i,j] = distance(v_l(s_i), v_l(s_j)), \quad i,j \leq m \tag{6}$$

$$corre[v_l(s_i), v_l(s_j)] = 1 - \frac{Cov(v_l(s_i), v_l(s_j))}{\sqrt{D(v_l(s_i))}\sqrt{D(v_l(s_j))}} \tag{7}$$

$$cos[v_l(s_i), v_l(s_j)] = \frac{v_l(s_i) \cdot v_l(s_j)}{|v_l(s_i)||v_l(s_j)|} \tag{8}$$

$$d[v_l(s_i), v_l(s_j)] = \sqrt{\sum_{o=1}^{O_l} (v_l(s_i)_o - v_l(s_j)_o)^2} \tag{9}$$

Third, the representational consistency rc_l of the lth layer in the global model and the local model can be calculated according to Formula (10), where u_l^{glo} and u_l^{loc} are the upper triangle of the corresponding RDMs.

$$rc_l(u_l^{glo}, u_l^{loc}) = \rho^2_{u_l^{glo}, u_l^{loc}} = (\frac{Cov(u_l^{glo}, u_l^{loc})}{\sigma_{u_l^{glo}} \sigma_{u_l^{loc}}})^2 \tag{10}$$

Finally, a layer-wise aggregation is implemented according to the representational consistency rc. Accordingly, the layers of local models are aggregated separately by using rc as the adaptive weight (representing the informative richness of a layer) to form the global models. The evaluation of RCE is summarized in Section 4.3.

3.4. Fed2A: The Integrated Form

The integrated form of Fed2A utilizes the uploading strategy PLU, and two aggregation strategies TVW and RCE simultaneously to achieve optimal performance. Specifically, Algorithm 1 shows the workflow of Fed2A in the integrated form. First, AFL clients are initialized by PLU, which defines the uploading frequency of deep and shallow layers. Second, required parameters such as ω_k , t_k and n_k are computed and then transmitted from client k to the collaborator. Third, the two adaptive weights in TVW and RCE of client k will be computed by the collaborator based on received parameters. Finally, the collaborator will calculate the adaptive weight $\overline{TW\_rc_{k,t}^l}$ for each layer and use it to update the global model. Note that the computation of adaptive weights requires no sensitive information, hence, user privacy can be guaranteed. The performance of Fed2A is evaluated in Section 4.4.

Algorithm 1 Fed2A: Integrated with *PLU*, *TVW*, and *RCE*.

Initialization: $PLU(R, P, D)$ are defined.
In a total of R rounds, there are $\frac{R}{P}$ periods;
In each period:
(1) Shallow layers are uploaded in all P rounds;
(2) Deep layers are uploaded in the last D rounds.

PART 1: : Executed in each AFL client
for each client $k \in K$ in parallel **do**
  ω_k is trained based on local data
  $t_k \leftarrow$ *current timestamp of the* k^{th} *client*
  $n_k \leftarrow$ *data size of the* k^{th} *client*
end for
Uploading ω_k, t_k and n_k to the collaborator
PART 2: : Executed in the AFL collaborator
Receiving ω_k, t_k, and n_k unblocking
while The trigger for a global aggregation is true **do** ▷ e.g., K local models are received.
  $t \leftarrow$ current timestamp, as the current global round
  Calculating $TW_{k,t}$ according to Formula (4)
  for $k \in K$ **do**
    for $l \in L$ **do**
      $rc^l_{k,t}$ calculated by Formula (10)
      $TW\_rc^l_{k,t} \leftarrow TW_{k,t} \times rc^l_{k,t}$
    end for
  end for
  for $k \in K$ **do**
    for $l \in L$ **do**
      $\overline{TW\_rc^l_{k,t}} \leftarrow \frac{TW\_rc^l_{k,t}}{\sum_{k=1}^{K}(TW\_rc^l_{k,t})}$ ▷ normalization
    end for
  end for
  for $l \in L$ **do**
    $w^l_{t+1} \leftarrow \sum_{k=1}^{K}(\overline{TW\_rc^l_{k,t}} \times w^l_k)$ ▷ $w^l_k \in w_k$.
  end for
end while

4. Evaluation and Discussion

In this section, first, common settings, such as used datasets, model to be trained, selected baselines, and evaluation metrics, are discussed. Second, the effectiveness and efficiency of PLU, TVW, RCE are evaluated respectively. Finally, the overall performance of Fed2A is revealed.

4.1. Common Settings

First, as shown in Figure 4, 30 clients are virtualized and deployed to participate in the learning process to learn a CNN model. Specifically, the CNN model is designed with two stacked convolution layers and two fully connected layers, and trained separately based on three standard datasets. Detailed training configuration for each dataset is listed as below:

- **FMNIST** (Fashion MNIST; https://github.com/zalandoresearch/fashion-mnist; (accessed on 1 February 2022) comprises 60,000 training images and 10,000 testing images with 10 labels. The data size of each image is $28 \times 28 \times 1$. To reflect the non-IID and unbalanced data distribution, each client possesses 1500–2500 training samples with 2–6 classes. The two convolution layers have 64 and 128 channels, respectively, followed by a 2×2 max-pooling layer. The two fully connected layers have 256 and 512 units, respectively, followed by a softmax unit as the output layer.

- **CIFAR-10** (https://www.cs.toronto.edu/kriz/cifar.html; (accessed on 1 February 2022) contains 50,000 training samples and 10,000 testing samples with 10 labels. Each sample is a $32 \times 32 \times 3$ color image. While using this dataset, each client is assigned with 1600–2400 training images in 2–6 classes. Since CIFAR-10 is a more complicated dataset, the corresponding CNN layers have doubled channels compared with FMNIST, and other layers are the same as FMNIST.
- **GermanTS** (https://bitbucket.org/jadslim/german-traffic-signs; (accessed on 1 February 2022) (German Traffic Signs Dataset) consists of 34,799 training pictures, 12,630 testing pictures, and 43 classes. The size of each picture is $32 \times 32 \times 3$. Moreover, 30 data partitions, each of which owns 900–1500 samples with 4–12 classes, are created and assigned to 30 clients respectively and not repeatedly. Since the label number is large, the corresponding CNN layers have half channels compared with FMNIST to reduce model parameters, and other layers are the same as FMNIST.

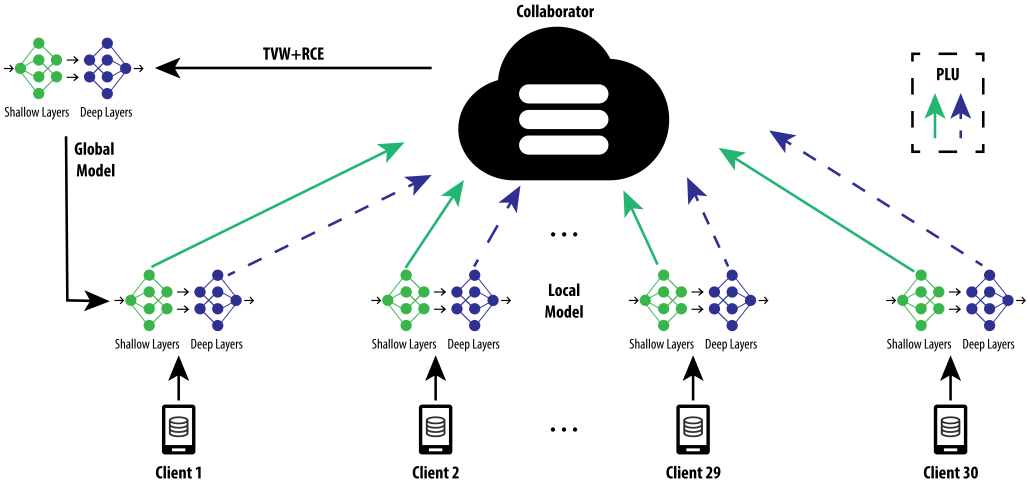

Figure 4. The deployment architecture of Fed2A.

Moreover, three state-of-the-art methods are used as the baselines, namely:

- **FedAvg** [1]: The most popular FL method. It randomly selects a fraction of clients as participants in a global round and aggregates local models averagely according to data size.
- **FedProx** [29]: A variant of FedAvg. It introduces a proximal term μ in the objective function to limit local changes for heterogeneous environments. Note that the value of μ used in this paper is 1.
- **FedAsync** [7]: A classic asynchronous FL algorithm. It includes a proximal term similar to FedProx. Since the collaborator immediately updates the global model whenever it receives a local model, a mixed hyperparameter α is designed in the aggregation process to balance the weights of the current global model and local model. Note that we use FedAsync + Poly with $a = 0.5$ and $\mu = 1$ in this paper.

Finally, three indicators are used as the evaluation metrics, namely:

- $\mathbf{I_1}$: The test accuracy of the global model in the final (300th) global round, which is defined according to Formula (11) (where TP, TN, FP, FN represent true positives, true negatives, false positives, false negatives, respectively). Note that the testing samples of each dataset are untouched during the FL learning process.
- $\mathbf{I_2}$: The global round number when the test accuracy of the global model first reaches the target accuracy. The target accuracy $T_{accuracy}$ is 65.00%, 35.00%, and 85.00% for FMNIST, CIFAR-10, and GermanTS, respectively, based on the performance of the three baselines.

- $\mathbf{I_3}$: The total communication cost when the target accuracy is reached, which is calculated according to Formula (12). Note that the estimated unit costs of shallow layers and deep layers for the client–server communication are summarized in Table 3.

$$acc = \frac{TP + TN}{TP + FP + FN + TN} \tag{11}$$

$$cost_{all} = \sum_{t=1}^{T_{target}} (cost^t_{shallow} + cost^t_{deep})$$

$$\begin{cases} cost_{shallow} = \frac{4 \times shallow_n}{1024 \times 1024} \\ cost_{deep} = \frac{4 \times deep_n}{1024 \times 1024} \end{cases} \tag{12}$$

Table 3. The estimated unit cost of communication.

Cost Type	Parameter Size	Estimated Cost
$cost_{shallow}$ for FMNIST	206,592	0.79 MB
$cost_{deep}$ for FMNIST	3,413,770	13.02 MB
$cost_{shallow}$ for CIFAR-10	829,184	3.16 MB
$cost_{deep}$ for CIFAR-10	9,574,154	36.52 MB
$cost_{shallow}$ for GermanTS	209,792	0.80 MB
$cost_{deep}$ for GermanTS	2,403,499	9.17 MB

4.2. Evaluation of PLU

To examine the influence of the most important parameters in PLU, i.e., P and D, ten PLU variants with $R = 300$, $P = 10$, and $D = 1$ *to* 10, are created. Note that $PLU(300, 10, 10)$ means that no optimization is made.

According to Table 4 "Uploading" Group, as the value of D decreases, the accuracy first increases and then decreases gradually. However, the round number when $T_{accuracy}$ is reached shows a consistent increase. It indicates that (1) uploading deep layers at the proper time can not only reduce communication cost but also improve model performance, and (2) when the uploading frequency of deep layers becomes too low, the accuracy and training speed will be affected due to the missing of vital information.

Moreover, for FMNIST, $PLU(300, 10, 7)$, which surpasses FedAvg and FedAsync, can achieve the highest accuracy of 70.03% and reach the target accuracy soonest at 88th round among all the PLU variants. In the meanwhile, PLU with $D = 7$ is a little worse than FedProx, however, its communication cost is 84.6% of the one of FedProx. As for CIFAR-10 and GermanTS, PLU with $D = 8$ outperforms all three baselines.

The above analysis shows that besides the optimization of the client-server communication, PLU can, surprisingly, improve the model accuracy. It proves that the separated training and uploading of shallow and deep layers of DNNs can efficiently and effectively reduce the consumption of limited local computing resources and also bring additional improvements in model accuracy for AFL.

Table 4. Summary of experiment results.

Group	Strategy	Accuracy at 300th Round			Round Reached $T_{Accuracy}$			$Cost_{target}$ (GB)		
		FMNIST	CIFAR-10	GermanTS	FMNIST	CIFAR-10	GermanTS	FMNIST	CIFAR-10	GermanTS
Baseline	*FedAvg*	69.91%	38.08%	**86.30%**	94	97	**265**	1.27	3.76	**2.58**
	FedProx	**70.22%**	**38.58%**	70.63%	**77**	**89**	-	**1.04**	**3.45**	-
	FedAsync	66.98%	38.27%	69.57%	95	132	-	1.28	5.12	-
Uploading	$PLU(300,10,1)$	66.80%	36.01%	84.31%	168	150	-	**0.44**	**1.32**	-
	$PLU(300,10,2)$	67.52%	35.54%	86.06%	120	119	269	0.50	1.47	**0.76**
	$PLU(300,10,3)$	68.92%	36.94%	85.68%	118	118	258	0.61	1.83	0.94
	$PLU(300,10,4)$	69.14%	37.34%	86.66%	99	101	257	0.65	1.95	1.16
	$PLU(300,10,5)$	69.46%	37.37%	87.50%	107	100	272	0.81	2.27	1.47
	$PLU(300,10,6)$	69.56%	38.26%	84.59%	90	97	285	0.81	2.48	1.78
	$PLU(300,10,7)$	**70.03%**	38.28%	87.07%	**88**	97	209	0.88	2.80	1.50
	$PLU(300,10,8)$	69.95%	**38.64%**	**87.57%**	94	**86**	**172**	1.04	2.76	1.37
	$PLU(300,10,9)$	69.99%	38.24%	86.37%	**88**	88	228	1.09	3.13	2.03
Aggregating	*TVW-exp*	69.32%	44.07%	89.06%	44	62	**133**	0.59	2.40	**1.29**
	TVW-inv	**71.70%**	**44.94%**	**89.62%**	**39**	**60**	**133**	**0.53**	**2.33**	**1.29**
	TVW-log	71.38%	41.54%	88.48%	44	66	144	0.59	2.56	1.40
	RCE	71.44%	41.07%	87.30%	51	84	200	0.69	3.26	1.95
Integrated	*Fed2A* *	**74.76%**	**45.99%**	**90.59%**	**18**	**54**	**120**	**0.20**	**1.74**	**0.97**

* The default PLUs of Fed2A are $PLU(300,10,7)$, $PLU(300,10,8)$ and $PLU(300,10,8)$ for FMNIST, CIFAR-10, and GermanTS, respectively. Moreover, the default TVW of Fed2A is *TVW-inv* for all the three datasets.

4.3. Evaluation of TVW and RCE

First, as summarized in Table 4 "Aggregating" Group, the three TVW variants, i.e., *TVW-inv*, *TVW-exp*, and *TVW-log*, can improve accuracy and learning speed significantly, and *TVW-inv* outperforms the other two variants in all three datasets. Specifically, against the best baselines in the three datasets (i.e., FedProx in FMNIST and CIFAR-10, and FedAvg in GermanTS), *TVW-inv* can achieve the highest accuracy of 71.70%, 44.94%, and 89.62% with an improvement of 2.11%, 16.49%, and 3.85%, respectively, and also first reach the target accuracy at 39th, 60th, and 133th round with an acceleration of 49.35%, 32.58%, and 49.81%.

Second, to evaluate RCE, in FMNIST and CIFAR-10, 50 stimuli (5 stimuli from each of the 10 categories) are randomly chosen from the testing dataset. Hence, the two RDMs will maintain a size of 50×50, and contain 1225 unique distance estimates. As for GermanTS, whose category number is 43, 86 stimuli (2×43) are used for representation, and the RDM has 3655 unique pairwise distances. We use cosine distance for computation convenience. As summarized in Table 4 "Aggregating" Group, an improvement in the test accuracy and a boost of learning speed ranging from 1.16% to 6.45%, and 5.62% to 33.77% are observed respectively by using RCE. It shows that the layer-wise aggregation based on the representational consistency can, indeed, enhance the learning performance.

Finally, as illustrated by Figure 5, although the three TVW variants can achieve higher test accuracy and reach the target accuracy faster than RCE, RCE can maintain a more stable learning curve in all three datasets. It indicates that the informative weight can have a steady improvement on the global model compared to the temporal weight.

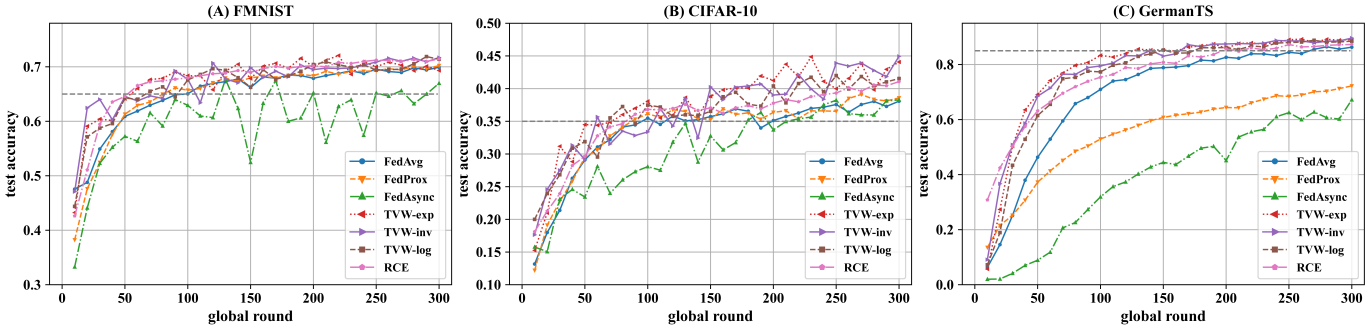

Figure 5. The accuracy curve of TVW and RCE compared with the baselines in (**A**) FMNIST; (**B**) CIFAR-10; and (**C**) GermanTS.

4.4. Evaluation of Fed2A

By utilizing PLU, TVW, and RCE together, Fed2A can achieve the optimal performance, namely:

1. As summarized in Table 4 "Integrated" Group, Fed2A can achieve the highest test accuracy about 75% in FMNIST, 46% in CIFAR-10, and 91% in GermanTS with a maximum increase of about 19%. Moreover, Fed2A can reach the $T_{accuracy}$ with a boost of about 76.62%, 39.36%, and 54.72%, compared with the best baseline in FMNIST, CIFAR-10, and GermanTS, respectively;
2. As illustrated in Figure 6, within each of the three datasets, Fed2A can maintain a sharper and stable accuracy curve and also outperform the three baselines consistently throughout the learning process;
3. As shown in Figure 7, Fed2A can reduce communication cost significantly by about 77.30% for FMNIST, 39.57% for CIFAR-10, and 62.40% for GermanTS, respectively.

In summary, the above evaluation results reveal the efficiency and effectiveness of Fed2A in supporting AFL by implementing a two-stage asynchronous learning procedure and three adaptive strategies, i.e., PLU, TVW, and RCE.

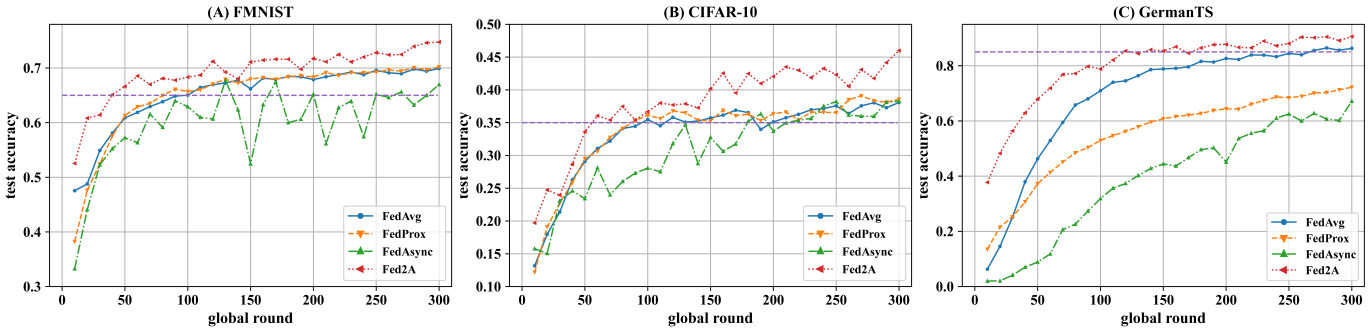

Figure 6. The accuracy curve of Fed2A compared with the baselines in (**A**) FMNIST; (**B**) CIFAR-10; and (**C**) GermanTS.

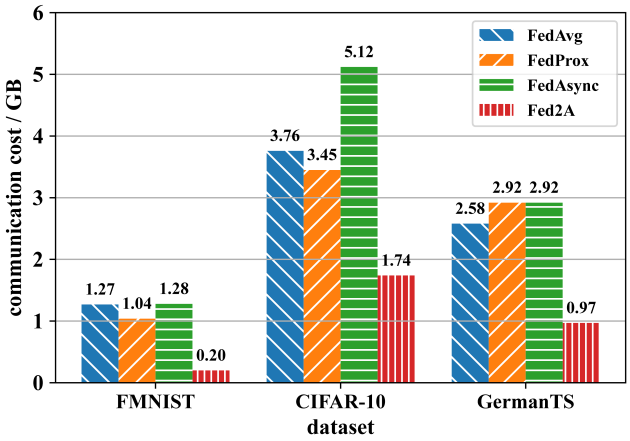

Figure 7. The communication cost of Fed2A compared with the baselines in the three datasets. Note that since FedProx and FedAsync have not reached the target accuracy for GermanTS, the corresponding costs are the overall costs of the total 300 rounds, i.e., 2.92 GB.

4.5. Discussion

First, the results of PLU indicate that deep layers contain much more redundant information compared with shallow layers, which not only wastes communication resources, but also influences model aggregation performance. Hence, reducing the uploading frequency of deep layers is beneficial to be communication-efficient. Moreover, for complicated datasets, such as CIFAR-10 and GermanTS, the optimal value of D is larger than simpler datasets such as FMNIST, which suggests that deep layers are more valuable for complex tasks and can be uploaded more frequently in these situations.

Second, *TVW-inv* performs the best among the three TVW variants, which shows that punishing the stale models harshly can enhance the aggregation process and newer parameters should be valued. The great performance improvement of TVW also indicates that we should pay more attention to the temporal heterogeneity issue of AFL in the future.

Third, the stable and satisfactory performance of RCE suggests the representational consistency presents the deep difference between local layers and global layers. It can quantify the contribution of each local layer and hence can steer the aggregation direction effectively and efficiently.

Finally, Fed2A, which integrates the three strategies, can achieve optimal performance in terms of test accuracy, training speed, and communication efficiency. It shows that Fed2A can tackle temporal and informative heterogeneity issues of AFL jointly, as well as saving communication resources significantly.

5. Conclusions

This paper introduces a federated learning mechanism in asynchronous and adaptive modes, called Fed2A. In Fed2A, both clients and the collaborator can run in an unblocking manner for local model uploading and global model aggregation. Moreover, to assist the asynchronous learning procedure, three adaptive strategies are proposed, namely, (1) PLU to adaptively upload shallow and deep layers of DNN layers, and (2) TVW and RCE to enhance the global aggregation based on adaptive weights calculated from the temporal and informative attributes of local parameters.

As for the three strategies, they are evaluated separately, and related results show that (1) the adaptive uploading frequency of deep and shallow layers can save communication costs significantly and, surprisingly, improve the model accuracy slightly; (2) stale models need to be punished while newer parameters deserve larger weights in AFL to boost model performance; (3) the representational consistency between local layers and global layers based on RDM can be utilized in the aggregation to guide the right convergence direction of the global model.

Moreover, the effectiveness and efficiency of Fed2A utilizing the three strategies jointly are evaluated based on three standard datasets (i.e., FMNIST, CIFAR-10, and GermanTS). As a result, compared with FedAvg, FedProx, and FedAsync, Fed2A can significantly improve model accuracy, training speed, and communication efficiency by about 19%, 76.62%, and 77.30%, respectively.

Although Fed2A performs well in terms of the three metrics, there are three limitations to Fed2A: (1) Fed2A only considers the uploading stage of local models, and the downstream optimization remains undiscussed; (2) an extra computation cost is needed for Fed2A to enhance the aggregation process by computing RDM; and (3) there are other formats of delay functions to be explored. Accordingly, in the future, these limitations will be addressed, namely, (1) PLU to support both upstream and downstream of the communication channel will be designed to further optimize the interaction between clients and the collaborator; (2) the computation in RCE will be simplified to accelerate the model aggregation process, and in turn, improve the learning performance; (3) a general format of temporal function will be studied, and a greedy algorithm will be designed to find the optimal solution.

Author Contributions: Conceptualization, S.L. and L.Y.; methodology, S.L. and L.Y.; software, S.L.; validation, Q.C.; formal analysis, Q.C.; investigation, Q.C.; resources, L.Y.; data curation, S.L.; writing—original draft preparation, S.L.; writing—review and editing, Q.C. and L.Y.; visualization, S.L. and L.Y.; supervision, L.Y.; project administration, L.Y.; funding acquisition, L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China grant number 62002398.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Fed2A	Federated learning mechanism in Asynchronous and Adaptive modes
IoT	Internet of Things
FL	Federated Learning
DNN	Deep Neural Network
AFL	Asynchronous Federated Learning
PLU	Periodic Layer Uploading strategy
TVW	Time Variety Weighting strategy
RCE	Representational Consistency Enhancing strategy
Non-iid	Non-independent and identically distributed
RDM	Representational Dissimilarity Matrix

References

1. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics. PMLR, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
2. Shaheen, M.; Farooq, M.S.; Umer, T.; Kim, B.S. Applications of Federated Learning; Taxonomy, Challenges, and Research Trends. *Electronics* **2022**, *11*, 670. [CrossRef]
3. Pokhrel, S.R.; Choi, J. Federated learning with blockchain for autonomous vehicles: Analysis and design challenges. *IEEE Trans. Commun.* **2020**, *68*, 4734–4746. [CrossRef]
4. Sarma, K.V.; Harmon, S.; Sanford, T.; Roth, H.R.; Xu, Z.; Tetreault, J.; Xu, D.; Flores, M.G.; Raman, A.G.; Kulkarni, R.; et al. Federated learning improves site performance in multicenter deep learning without data sharing. *J. Am. Med. Inform. Assoc.* **2021**, *28*, 1259–1264. [CrossRef] [PubMed]
5. Qu, Y.; Pokhrel, S.R.; Garg, S.; Gao, L.; Xiang, Y. A blockchained federated learning framework for cognitive computing in industry 4.0 networks. *IEEE Trans. Ind. Inform.* **2020**, *17*, 2964–2973. [CrossRef]
6. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* **2021**, *14*, 1–210. [CrossRef]
7. Xie, C.; Koyejo, S.; Gupta, I. Asynchronous federated optimization. *arXiv* **2019**, arXiv:1903.03934.

8. Liu, Y.; Qu, Y.; Xu, C.; Hao, Z.; Gu, B. Blockchain-enabled asynchronous federated learning in edge computing. *Sensors* **2021**, *21*, 3335. [CrossRef] [PubMed]
9. Lu, Y.; Huang, X.; Dai, Y.; Maharjan, S.; Zhang, Y. Differentially private asynchronous federated learning for mobile edge computing in urban informatics. *IEEE Trans. Ind. Inform.* **2019**, *16*, 2134–2143. [CrossRef]
10. Chen, Y.; Ning, Y.; Slawski, M.; Rangwala, H. Asynchronous online federated learning for edge devices with non-iid data. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020; pp. 15–24.
11. Chen, Y.; Sun, X.; Jin, Y. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *31*, 4229–4238. [CrossRef]
12. Caldas, S.; Konečny, J.; McMahan, H.B.; Talwalkar, A. Expanding the reach of federated learning by reducing client resource requirements. *arXiv* **2018**, arXiv:1812.07210.
13. Wang, H.; Sievert, S.; Liu, S.; Charles, Z.; Papailiopoulos, D.; Wright, S. Atomo: Communication-efficient learning via atomic sparsification. *arXiv* **2018**, arXiv:1806.04090v3.
14. Sattler, F.; Wiedemann, S.; Müller, K.R.; Samek, W. Robust and communication-efficient federated learning from non-iid data. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *31*, 3400–3413. [CrossRef]
15. Asad, M.; Moustafa, A.; Ito, T. FedOpt: Towards communication efficiency and privacy preservation in federated learning. *Appl. Sci.* **2020**, *10*, 2864. [CrossRef]
16. Lu, X.; Liao, Y.; Lio, P.; Hui, P. Privacy-preserving asynchronous federated learning mechanism for edge network computing. *IEEE Access* **2020**, *8*, 48970–48981. [CrossRef]
17. Yu, H.; Yang, S.; Zhu, S. Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 5693–5700.
18. Huang, L.; Yin, Y.; Fu, Z.; Zhang, S.; Deng, H.; Liu, D. LoAdaBoost: Loss-based AdaBoost federated machine learning with reduced computational complexity on IID and non-IID intensive care data. *PLoS ONE* **2020**, *15*, e0230706. [CrossRef]
19. Wang, J.; Joshi, G. Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD. *arXiv* **2018**, arXiv:1810.08313
20. Wang, S.; Tuor, T.; Salonidis, T.; Leung, K.K.; Makaya, C.; He, T.; Chan, K. Adaptive federated learning in resource constrained edge computing systems. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1205–1221. [CrossRef]
21. Kang, D.; Ahn, C.W. Communication Cost Reduction with Partial Structure in Federated Learning. *Electronics* **2021**, *10*, 2081. [CrossRef]
22. Wang, Q.; Li, Q.; Wang, K.; Wang, H.; Zeng, P. Efficient federated learning for fault diagnosis in industrial cloud-edge computing. *Computing* **2021**, *103*, 2319–2337. [CrossRef]
23. Wang, Y.; Kantarci, B. Reputation-enabled federated learning model aggregation in mobile platforms. In Proceedings of the ICC 2021-IEEE International Conference on Communications, Montreal, QC, Canada, 14–23 June 2021; pp. 1–6.
24. Chen, S.; Shen, C.; Zhang, L.; Tang, Y. Dynamic aggregation for heterogeneous quantization in federated learning. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 6804–6819. [CrossRef]
25. Wang, X.; Li, R.; Wang, C.; Li, X.; Taleb, T.; Leung, V.C. Attention-Weighted Federated Deep Reinforcement Learning for Device-to-Device Assisted Heterogeneous Collaborative Edge Caching. *IEEE J. Sel. Areas Commun.* **2020**, *39*, 154–169. [CrossRef]
26. Li, S.; Ngai, E.; Ye, F.; Voigt, T. Auto-weighted Robust Federated Learning with Corrupted Data Sources. *arXiv* **2021**, arXiv:2101.05880.
27. Raghu, M.; Gilmer, J.; Yosinski, J.; Sohl-Dickstein, J. SVCCA: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 6078–6087.
28. Mehrer, J.; Spoerer, C.J.; Kriegeskorte, N.; Kietzmann, T.C. Individual differences among deep neural network models. *Nat. Commun.* **2020**, *11*, 1–12. [CrossRef] [PubMed]
29. Li, T.; Sahu, A.K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; Smith, V. Federated optimization in heterogeneous networks. In Proceedings of the Machine Learning and Systems, Austin, TX, USA, 2–4 March 2020; pp. 429–450.