electronics

MDPI

Article

A Clustered Federated Learning Method of User Behavior Analysis Based on Non-IID Data

Jianfei Zhang * and Zhongxin Li

School of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130000, China
* Correspondence: jfzhang@cust.edu.cn

Abstract: Federated learning (FL) is a novel distributed machine learning paradigm. It can protect data privacy in distributed machine learning. Hence, FL provides new ideas for user behavior analysis. User behavior analysis can be modeled using multiple data sources. However, differences between different data sources can lead to different data distributions, i.e., non-identically and non-independently distributed (Non-IID). Non-IID data usually introduce bias in the training process of FL models, which will affect the model accuracy and convergence speed. In this paper, a new federated learning algorithm is proposed to mitigate the impact of Non-IID data on the model, named federated learning with a two-tier caching mechanism (FedTCM). First, FedTCM clustered similar clients based on their data distribution. Clustering reduces the extent of Non-IID between clients in a cluster. Second, FedTCM uses asynchronous communication methods to alleviate the problem of inconsistent computation speed across different clients. Finally, FedTCM sets up a two-tier caching mechanism on the server for mitigating the Non-IID data between different clusters. In multiple simulated datasets, compared to the method without the federated framework, the FedTCM is maximum 15.8% higher than it and average 12.6% higher than it. Compared to the typical federated method FedAvg, the accuracy of FedTCM is maximum 2.3% higher than it and average 1.6% higher than it. Additionally, FedTCM achieves more excellent communication performance than FedAvg.

Keywords: federated learning; Non-IID; user behavior; user modeling

Citation: Zhang, J.; Li, Z. A Clustered Federated Learning Method of User Behavior Analysis Based on Non-IID Data. *Electronics* **2023**, *12*, 1660. https://doi.org/10.3390/electronics12071660

Academic Editor: Antoni Morell

Received: 27 February 2023
Revised: 24 March 2023
Accepted: 28 March 2023
Published: 31 March 2023

1. Introduction

As the computer field boomed, users generated a variety of behavior data while surfing the Internet, such as video-clicking behavior, shopping behavior, and more. In recent years, deep learning techniques have been used to uncover the hidden information behind such behavioral data. It is well known that the predictive power of deep learning models relies on training data. However, with the increasing emphasis on user privacy, it will be more difficult to collect and share data across organizations, thus creating isolated data islands. Besides, the owners of some highly sensitive data may object to the unrestricted use of such data. In this environment, how to solve the fragmented data and isolated data island will become the primary problem in the field of machine learning.

To simultaneously achieve privacy protection and train models using data, FL is proposed. FL aims to build a federated model based on global data. Edge devices with fragmented data participate in training models using their data while keeping their data secure. In this case, the client does not transmit their local data but rather the parameters of the model trained using the local data. At the end of the training, all clients will obtain a model that meets the requirements.

However, user behavior data is influenced by age, gender, lifestyle, and other factors. The local data on the client side is likely to be non-independently and non-identically distributed. Due to the characteristic of "data does not move, model moves" of FL, the central server cannot directly operate the local data of each client. The authors of [1] proposed a

framework for FL and designed the FedAvg algorithm. The FedAvg algorithm has excellent performance on identically and independently distributed (IID) data. However, Non-IID data can affect the performance of the FedAvg algorithm. As the degree of Non-IID increases, there will be a significant decrease in model accuracy. Local training of FL relies on stochastic gradient descent, and only the IID of local data ensures that the local stochastic gradient is an unbiased estimate of the global gradient. Gradient differences will cause the global model to be influenced by the client-side local model. Hence, differences in data distribution between clients are a critical issue for federated learning.

Recently, various approaches attempting to solve the impact of Non-IID data on FL models have been proposed. A part of the method applies adaptive optimization. In [2–4], these methods limit the update magnitude of the local model by adding proximal terms and reducing the gap between the local model and the global model. However, all these methods require adjusting the ratio parameter of the proximal term. Different ratio parameters can affect the accuracy and convergence speed of the model. A part of the approach tries to adjust the data distribution of all clients to improve the performance of the FL model. In [5,6], clients share a part of the data to build a public dataset. The public dataset can reduce data distribution differences across all clients. In [7,8], data augmentation is performed for classes with fewer samples to balance the data samples for each client. The method of sharing data will leak privacy, especially in the field of user behavior analysis. User data are supposed to be highly confidential, and sharing data methods violate the original intent of federated learning to protect privacy.

As the Internet grows, users may interact with different devices. User behavior data is stored on multiple devices. The devices involved in the training may be cell phones, tablets, or edge devices with restricted resources. As a result, during the training of the model, the devices have different computational speeds called system heterogeneity. In [1–3], the client and server use synchronous communication, which has the advantage of high accuracy and fast convergence. Since clients have different computational speeds, if synchronous communication is used, the device with the slowest computation speed will prolong the overall training time. The existing approach [2] is to drop the slow training devices, but dropping the slow devices only alleviates the problem of 'wait'. Randomly dropping devices will destroy the integrity of the overall data. Asynchronous communication solves the problem of different computing speeds for clients. It requires all cluster mediators to communicate directly with the server after the local update is complete rather than waiting for all clusters. Due to the special way of asynchronous communication, each client can only generate aggregated models with the server individually. However, the aggregation parameter weights are difficult to adjust. A series of methods for aggregating parameter weights for asynchronous communication was proposed in [9,10], but these methods still have gaps with synchronous communication.

We propose an FL algorithm based on a two-tier cache mechanism (FedTCM) that supports the mitigation of the impact of Non-IID data on the model and considers the different computational speeds of devices. First, FedTCM clusters all clients based on the similarity of their data distribution, and the clients within the cluster can optimize their model parameters. Additionally, this process does not involve operating the private data of the client, so it is more secure. Second, due to the difference in the computing speed of the devices, the cluster mediator asynchronously communicates with the server. Finally, FedTCM sets up a two-tier cache structure on the server side, where the first-tier cache stores the latest model parameters for each cluster. In this way, FedTCM avoids the problem of difficult adjustment of aggregation weights for traditional asynchronous communication. In the second-tier cache, we use a random distribution strategy to train the model in different clusters. If the training order is not considered, the model in the second-tier cache will contain data information for all clusters. The impact of Non-IID data on FL models is reduced by a random distribution strategy.

The main contributions of this paper are summarized as follows:

- To solve the problem of the impact of Non-IID data on the FL model and the difficulty in adjusting parameters due to asynchronous communication, FedTCM sets up a two-tier cache structure in the server, which results in accuracy improvement in non-IID environments.
- Using intra-cluster synchronous communication and inter-cluster asynchronous communication mitigates the impact of varying client computation speed and reduces the communication burden on the server.
- The performance evaluation on the user behavior dataset shows that the algorithm in this paper has high accuracy compared to existing algorithms.

2. Related Work

To solve the impact of Non-IID data, various methods have been proposed. The existing work falls into four main categories: data augmentation, cluster-based multi-model learning, adaptive optimization methods, and personalized federated learning.

2.1. Data Augmentation

The authors of [5] found that the difference between local and global data distribution reduces the accuracy of the model and proposed an approach based on data sharing. Experiments show that sharing 5% of the data per client will improve accuracy by 30%. The literature [7] uses the smote oversampling technique to generate new samples for categories with a small number of samples. Similarly, the authors of [11] use the conditional generative network to generate new samples. These methods reduce the heterogeneity of local client data by way of data expansion. However, these methods involve operating client local data, which violates the original intent of FL to protect privacy.

2.2. Cluster-Based Multi-Model Learning

The literature [12] proposed an approach based on multiple centers. The local model and center location are iteratively optimized by modifying the loss function of the local client. Clients with similar distances to the same center are divided into groups. The clients in the group train the same model. The authors of [13] propose a novel clustering approach where the process of clustering does not require the participation of all clients to avoid the additional communication overhead added during clustering. The literature [14] proposed FedAMP, an attention-based FL algorithm. Different from the previous approach, FedAMP does not put the client into a specified certain cluster. By introducing the attention mechanism, a client can maintain multiple cluster models at the same time. These methods can perform well in Non-IID; clients with different data distributions do not interfere with each other. However, in a real-world application setting, multiple models do not generalize for all clients. Our goal is to train an FL model that is shared by all clients.

2.3. Adaptive Optimization

The literature [2] proposed adding proximal terms on the client side. The proximal term is used to limit the gap between the global model and the local models. The authors of [3] proposed an FL framework based on gradient correction. The gradient correction term is used to limit the extent of model updates. The study [4] proposed to estimate the global device knowledge separately using local control variables and server control variables. The literature [15] found that when only a small number of devices are involved in the training, the model accuracy decreases significantly; they also proposed that the momentum-based algorithm FedAvgM improves the accuracy. FedAvgM speeds up the model training process and improves model accuracy by adjusting the gradient. These methods perform well in improving FL model convergence. However, these methods are difficult to adjust the ratio parameter of the proximal term.

2.4. Personalized Federated Learning

The literature [16] introduced the concept of life-long learning. This method treats clients with different distributions as different tasks. After training several rounds, all the tasks are combined into a global task. The authors of [17] introduced the FL of taskonomy to efficiently aggregate heterogeneous data by learning task correlations between clients. The study [18] proposed a delayed aggregation method. After the training of local clients is completed, the server collects all model parameters. Then, the server sends these model parameters to other clients for further training. This algorithm uses multiple clients to train the same model to mitigate the impact of heterogeneous data. The literature [19] proposed the ARUBA framework. It is based on online learning to implement adaptive meta-learning under FL. When used in combination with FedAvg, it can improve model generalization performance.

In this work, we propose an FL algorithm based on a two-tier caching mechanism called FedTCM. FedTCM clusters similar clients, and clients within the same cluster perform local training synchronously. Clusters send model parameters to the server asynchronously to overcome the heterogeneity of the system. On the server side, we propose a caching mechanism to solve the impact of heterogeneous data on the model. Therefore, FedTCM is a combination of centralized and decentralized federated learning methods.

3. The Principle of FedTCM

To reduce the impact of Non-IID data on the performance of the FL model, we design an FL algorithm based on a two-tier cache mechanism (FedTCM). The main parameters of the system are summarized in Table 1.

Table 1. Notion and definitions.

Notation	Definition
FL	federated learning
Non-IID	non-identically and non-independently distributed
m	first-tier cache list
$m[k]$	k^{th} position of list m
n	second-tier cache list
$n[k]$	k^{th} position of list n
V_i	the label count vector of clients i
$v_{i,h}$	the components of the vectors V_i
q	the cosine similarity between clients
l_k	loss function
d	the number of dataset labels
a_d	the one-hot vector of the model output
$\{x_k, y_k\}$	the characteristics and label of the data sample
y_d	the one-hot vector of y_k
F	the local empirical risk of the client
$P_{b,j}$	the dataset of client j in cluster b
n_b	the number of datasets in cluster b
$n_{b,j}$	the number of datasets of client j in cluster b
$\omega_{b,j}^{t+1}$	the model parameters of client j in cluster b at $t+1$
ω_b^{t+1}	the aggregation parameters of cluster b at $t+1$
W_b^{t+1}	the aggregation parameters of first-tier cache at $t+1$
η	learning rate
e	epoch
p	the threshold value of cosine similarity
C	all clusters
S	the number of clusters
N_a	the number of clients
α	the degree of heterogeneity of the data distribution
B	batch size
c_t	delay time parameter

Figure 1 shows the structure of FedTCM. Although we cannot rearrange the data for each client, we can rearrange each client. In FedTCM, we first cluster all the clients. The clustering is based on the similarity of the client data distribution. A model trained commonly by clients with similar data distribution is better than a model trained by a single client. Additionally, the degree of heterogeneity between clients in the cluster is reduced. Clients in the same cluster can optimize their models and reduce the number of communications with the server during the training process, ensuring system stability. After the clustering is complete, the server sends an initial model to each cluster. The cluster mediators share the initial model with each client within the cluster, and the local clients start training. Local clients send the model parameters to the cluster mediator after the training is complete. The cluster mediators aggregate model parameters of all clients within the cluster and send the model to the server through asynchronous communication. Asynchronous communication will effectively alleviate the problem of system heterogeneity.

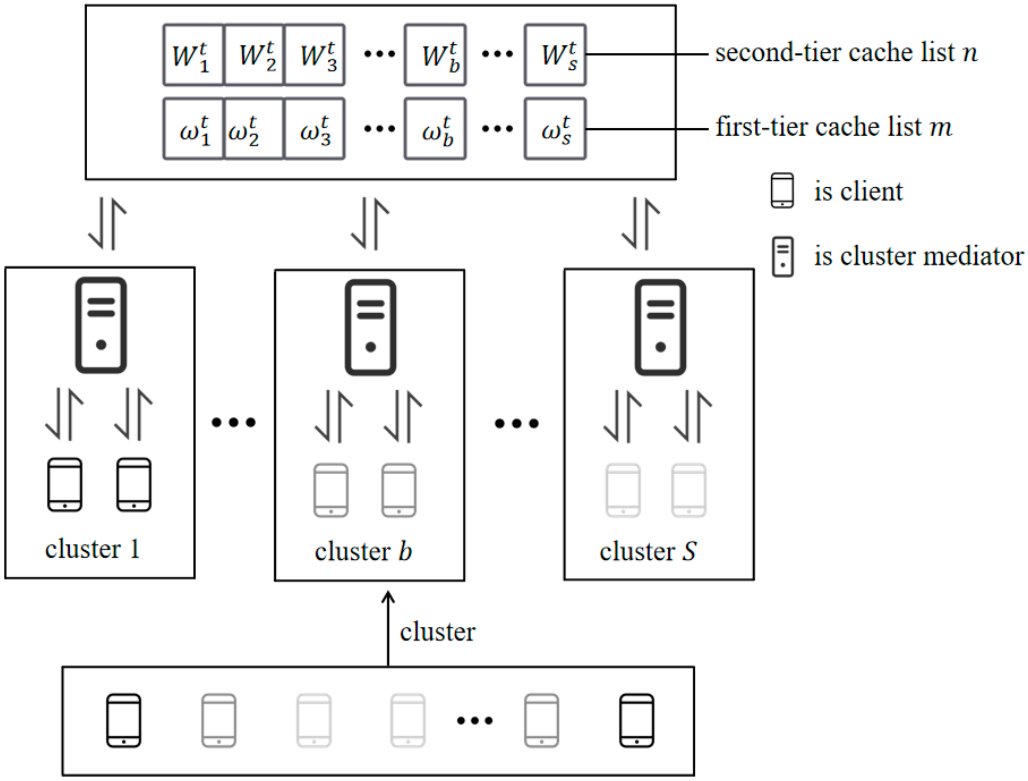

Figure 1. Structure of FedTCM. FedTCM mainly includes clusters and a server. Clients with different colors denote that the clients have different data distributions. The cluster consists of clients and a cluster mediator. The server contains a first-tier cache list m and a second-level cache list n.

At the same time, we design a two-tier caching mechanism on the server side. Each cluster corresponds to one location in the first-tier cache and one in the second-tier cache, respectively. When the cluster calculation is completed, the model is sent to the server. The server updates the corresponding locations in the first-tier cache and the second-tier cache. The first-tier cache is used to store the latest model for each cluster, and it is used to solve the problem of difficulty in adjusting aggregation weights in asynchronous communication. In traditional asynchronous communication, the server requires aggregation of a single model parameter with the model parameters in the server, but the aggregation weights affect the performance of the model. Since FedTCM set up the first-tier cache, after the cluster computation is completed, FedTCM only needs to aggregate all the models in the first-tier cache. In this way, FedTCM avoids aggregation of single model parameters with server model parameters. The second-tier cache stores the aggregated model parameters

corresponding to the first-tier cache, and it uses a model random distribution strategy to mitigate the impact of Non-IID data on the model. Inspired by the literature [18], we believe that a model trained on all the clustered data will be more reliable than a model from a single cluster if the order of training of the data is not considered. Therefore, FedTCM let the models in the second-tier cache be trained with data from different clusters. In this way, the models in the second-tier cache carry more information about the data of the cluster. The degree of model heterogeneity in the second-tier cache will be lower than that of the models trained in a single cluster. Therefore, the second-tier cache is used to mitigate the impact of Non-IID on the model.

3.1. Clustering the Clients Based on the Distribution of Local Data

Various existing methods [12,20–22] use the model or the computational speed of the device in the process of clustering. However, all these methods need to be pre-trained before clustering. We propose to use the number of sample labels from the client for clustering. The client counts the amount of local data before local training begins. Calculate the percentage of each type of label according to the number of labels in the sample data. We call it the label count vector. FedTCM cluster is based on the similarity of the label count vector. The approach has two advantages. One is that sending a label count vector to an honest server does not leak data from the client, and the other is that we do not need to pre-train before clustering compared to previous clustering methods. The label count vector is one-dimensional, and the size of the label count vector is negligible compared to the model. Before the training starts, clients send the local label count vector to the server. The server clusters all clients according to the cosine similarity of the label count vector. The cosine similarity is calculated using Equation (1).

$$q = \frac{V_i \cdot V_j}{||V_i|| ||V_j||} = \frac{\sum_{k=1}^{n} v_{i,h} \times v_{j,h}}{\sqrt{\sum_{h=1}^{n} (v_{i.h})^2} \times \sqrt{\sum_{h=1}^{n} \left(v_{j,h}\right)^2}}, \tag{1}$$

where V_i and V_j represent the label count vector of clients i and j, respectively. $v_{i,h}$ and $v_{j,h}$ represent the components of the vectors V_i and V_j, q denotes the cosine similarity between clients.

The server calculates the cosine similarity between all clients. If the cosine similarity of the label count vector of several clients exceeds a set threshold, then they are in the same cluster. Data within a cluster approximately conforms to the same data distribution.

3.2. The Process of Training on Clients

In the training process, the main tasks of the client are receiving the initial model from the cluster mediator, training the model using local data, and uploading the model parameters to the cluster mediator.

In the process of local client training, the target of the local clients is defined as follows:

$$minF = \frac{1}{n_{b,j}} \sum\nolimits_{k \in P_{b,j}} l_k(x_k, y_k; \omega), \tag{2}$$

where l_k is the loss function for data sample $\{x_k, y_k\}$, x_k is the characteristics of the data sample, and y_k is the label of the data sample. The subscript b, j represents the client j in cluster b. $P_{b,j}$ is the dataset of client j. Let $n_{b,j} = \left|P_{b,j}\right|$. F is the local empirical risk of the device in the local data $P_{b,j}$. The loss function l_k uses the cross-entropy loss function, which is defined as follows:

$$l_k = -\sum\nolimits_d y_d \log a_d, \tag{3}$$

where d is the number of dataset labels, y_d is the one-hot vector of y_k, and a_d is the one-hot vector of the model output. To minimize the loss in the client training process, the local client iteration phase (the stochastic gradient method as an example) is interpreted as:

$$\omega_{b,j}^{t+1} = W^t - \eta \nabla F, \tag{4}$$

where W^t is the model parameters sent by the cluster mediator to client j in round t, $\omega_{b,j}^{t+1}$ is the local model computed by the local client based on W^t, and η represents the learning rate. When the local client has finished computing, the local client sends the local model parameters to the cluster mediator.

3.3. The Major Tasks of Cluster Mediator

The cluster mediator has two main tasks in the process of model training. One task is to receive the initial model from the server and send the initial model to each client in the cluster. Another is to receive the model parameters of all clients in the cluster, aggregate the collected model parameters, and send the aggregated model parameters to the server. The cluster mediator collects the local models and aggregates them using Equation (5).

$$\omega_b^{t+1} = \sum \frac{n_{b,j}}{n_b} \omega_{b,j}^{t+1}, \tag{5}$$

where $\omega_{b,j}^{t+1}$ represents the model parameters for the client j in cluster b at round $t+1$, and ω_b^{t+1} represents the aggregation parameters of cluster b at round $t+1$. The cluster mediators send ω_b^{t+1} to the server after the aggregation is completed. Due to the difference in the computational speed of the client, the cluster mediator will communicate with the server asynchronously.

After the client communicates with the cluster mediator, the cluster mediator immediately aggregates all the received models. The data distribution of all clients in the same cluster is consistent. It avoids the interference of other clients when aggregating model parameters. The model parameters of the cluster mediator are more representative of the overall data distribution of the cluster. Only all cluster mediators need to communicate with the server to cover all client information, instead of all clients.

We measured the communication performance of FedTCM in terms of the number of times the parameter is sent and received. FedTCM puts clients with similar data distribution into the same cluster. The model information of a cluster mediator represents the data information of all clients within the cluster. The number of cluster mediators is much smaller than the number of clients. In addition, FedTCM uses asynchronous communication and the two-tier cache mechanism. When aggregating model parameters, the server does not need to communicate with all clients, but only with part of all cluster mediators to complete the calculation. Additionally, each cluster mediator only needs to communicate with clients in its cluster to complete the local computations. Therefore, with the two-tier caching mechanism and cluster mediator, FedTCM can reduce the number of times the server receives and sends model parameters.

3.4. The Major Tasks of the Server

We designed a two-tier caching mechanism on the server side. The first-tier cache is used to avoid the problem of adjusting the aggregation parameters for asynchronous communication. The second-tier cache uses a random distribution strategy to mitigate the impact of Non-IID data on the model.

In the previous section, we mentioned that the cluster communicates with the server using asynchronous communication methods. However, there is a disadvantage of asynchronous communication: aggregation weights are difficult to adjust when a single model is aggregated with a model stored on the server. To avoid the impact of aggregation

weights on the model in asynchronous communication, the first-tier cache is used to solve this problem.

The first-tier cache structure is shown in Figure 2. The cluster mediator of cluster b sends model parameter ω_b^{t+1} to the server. The server replaces the stale model parameters ω_b^t of cluster b with the latest model parameters ω_b^{t+1}, which means that the first-tier cache will store the latest models of all clusters. FedTCM uses Equation (6) to aggregate all model parameters in the first-tier cache and store these aggregation model parameters in the corresponding location of the second-tier cache.

$$W_b^{t+1} = \frac{1}{S}\sum_{i=1}^{S} m[i] \tag{6}$$

where S is the number of clusters, and m is the first-tier cache list. Although FedTCM uses asynchronous communication, it avoids aggregation of single model parameters with server model parameters.

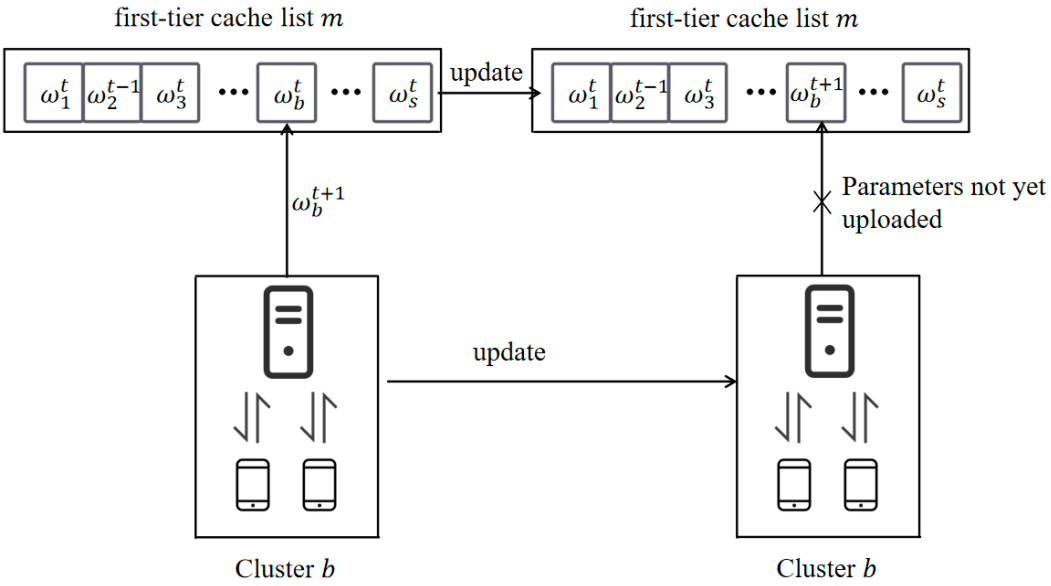

Figure 2. Update principle of the first-tier cache.

The second-tier cache uses a model random distribution strategy. Figure 3 illustrates the principle of the second-tier cache. After the first-tier cache aggregation is completed, the aggregation model parameter W_b^{t+1} is stored in the second-tier cache. The server will randomly select a model parameter in the second-tier cache to send to cluster b (the second-tier cache model of cluster S is selected in Figure 3). Cluster b receives the model parameters and starts the next round of training. We aggregate all the model parameters in the second-tier cache to obtain the global model. The global model does not participate in training, and it can be obtained at any time point. This approach allows the models in the second-tier cache to be trained based on different cluster data. Therefore, the model in the second-tier cache can carry more information about the data of the cluster. The random distribution strategy will help to reduce global loss. The model in the second-tier cache gradually approaches the optimal global model. The heterogeneity of the models in the second-tier cache is lower compared to the single cluster model. In this way, a simple aggregation of the model parameters in the second-tier cache can reduce the impact of Non-IID data on the model.

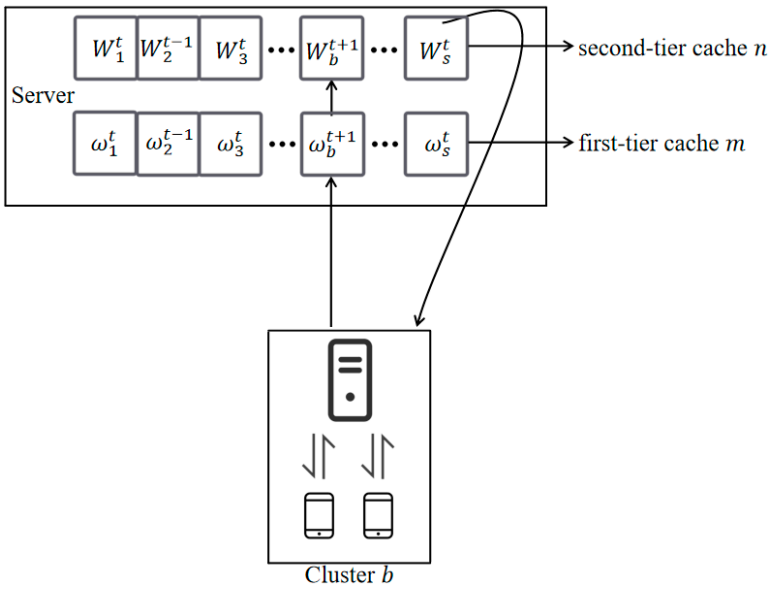

Figure 3. Update principle of the second-tier cache. The second-tier cache sends the model to cluster b using a random distribution.

3.5. The Process of FedTCM

FedTCM is illustrated in Figure 4. The clients in cluster b send the model parameters to the cluster mediator (step 1). The cluster mediator aggregates the received parameters. The cluster mediator sends the aggregated model parameters ω_b^{t+1} to the server. Asynchronous uploading is used here (step 2). The server updates the model parameters in the first-tier cache (step 3). The server aggregates all model parameters in the first-tier cache and stores the aggregated results W_b^{t+1} in the corresponding second-tier cache (step 4). The server will randomly select a model in the second-tier cache and send the model to cluster b (step 5). The cluster mediator sends the model to the client. Intuitively, cluster b is trained based on model parameters from different clusters (step 6).

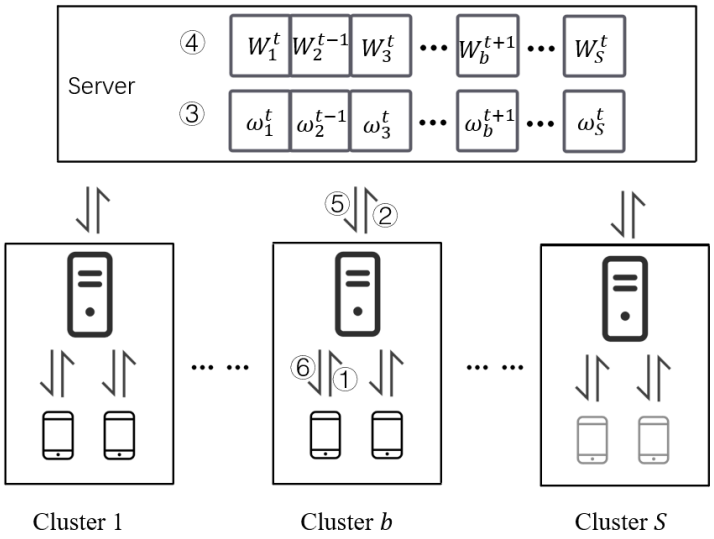

Figure 4. The process of FedTCM.

The pseudocode of FedTCM is shown in Algorithm 1. Line 3 represents our clustering method based on cosine similarity. Lines 6–7 represent the update and aggregation process of the first-tier cache, where m represents the list of first-tier cache, and m is used to store the latest model for each cluster. Lines 8–9 represent our random distribution strategy,

where n denotes the list of the second-tier cache, and n is used to store the aggregation model parameters from the first-tier cache. Lines 13–16 represent the process of receiving and sending models by cluster mediators. Lines 18–22 represent the training process of the local client.

Algorithm 1. FedTCM.

Input: $cluster()$ is the cosine similarity clustering algorithm
Output: global model g^t
1: **server process:**
2: Before training starts, receive label count vector v_i
3: $C = cluster(v_i)$
4: **for** t = 0,1 . . . ,T **do:**
5: Receive model ω_b^{t+1} from cluster b
6: $m[b] = \omega_b^{t+1}$
7: $W_b^{t+1} = \frac{1}{S}\sum_{i=0}^{S} m[i]$
8: $n[b] = W_b^{t+1}$
9: Send $n[k]$ to cluster b, $k \in random(0, S)$
10: **end for**
11: $g^t = \frac{1}{S}\sum_{i=0}^{S} n[i]$
12: **cluster mediator:**
13: Receive W^t from server, send W^t to clients in the cluster
14: Receive $\omega_{b,j}^{t+1}$ from clients in cluster
15: Aggregate the collected parameters: $\omega_b^{t+1} = \frac{n_{b,j}}{n_b}\omega_{b,j}^{t+1}$
16: Send ω_b^{t+1} to server
17: **client device:**
18: Receive W^t from cluster mediator
19: **for** local iteration **do:**
20: local update $\omega_{b,j}^{t+1} \Leftarrow W^t - \eta\nabla F$
21: **end for**
22: Send update model $\omega_{b,j}^{t+1}$ to cluster mediator

4. Experiment and Results

In this section, we introduce the experiments and analyze the simulation results to verify the performance of the FedTCM. In our experiments, there are N_a clients, S mediators, and one central server. The data similarity of the clients is used to divide the N_a clients into different clusters. S mediators will be chosen randomly by the clients in each cluster. We present the results of our experiments on user shopping behavior data and user sports behavior data.

For every experiment on user shopping behavior data, we repeated the following hyperparameters. SGD was used as the optimization method (learning rate, $\eta = 0.1$; epoch, $e = 5$; and the number of clients participating in training, $N_a = 20$). Cosine similarity is used to describe the degree of similarity of client data. The threshold value of cosine similarity is $p = 0.98$.

For every experiment on user sports behavior data, we repeated the following hyperparameters. SGD was used as the optimization method (learning rate, $\eta = 0.03$; epoch, $e = 5$; batch size, $B = 3$; and the number of clients, $N_a = 10$). The threshold value of cosine similarity is $p = 0.98$.

4.1. Dataset and Pre-Processing

In recent years, online shopping has become the most convenient way to shop. The huge amount of user behavior data can support various large training tasks. Exploring the hidden information can reduce the recommendation cost of an e-commerce platform, and it will provide a great convenience for online shopping in practical applications. The data we selected contains multiple user characteristics and shopping behaviors. After removing the null and error values in the dataset, we select 10 categories of items. This dataset is

highly heterogeneous and non-uniform. In this dataset, users have four types of behavior: browse, like, add to cart, and buy. We need to use known features to forecast the behavior of the user.

Moreover, we evaluated the results of our experiments on the user sports behavior dataset. The smart devices that users carry around with them contain various sensors. It does not require active user settings and can record various sports behaviors (including walking, running, etc.). The data we selected contains a variety of user characteristics and exercise habits, so we will use the user sports behavior data to predict the user's physical fitness. The physical condition of the user reflects the lifestyle habits of the user; smart devices can send exercise reminders and customize personalized exercise programs to users based on their physical fitness. Hence, user sports behavior analysis will be of relevance.

4.2. Federated Data Splitting

The goal of FedTCM is to mitigate the impact of Non-IID data on the model. Therefore, we will set up Non-IID data to represent this problem when dividing the dataset. We use the Dirichlet distribution to generate Non-IID data for each client. The Dirichlet distribution is also known as the multivariate Beta distribution. The density function of Dirichlet is Equation (7):

$$Dir(X|\alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^{d} X_i^{\alpha_i - 1} \tag{7}$$

where $\alpha = \{\alpha_1, \ldots, \alpha_d\} > 0$, $B(\alpha) = \frac{\prod_{i=1}^{d} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{d} \alpha_i\right)}$. We can sample the dataset according to the Dirichlet distribution. The parameter α of the Dirichlet distribution can control the sampling probability of each class of labels in the dataset. In this way, the data amount of each client can calculate the number of each label.

4.3. Baseline Algorithm

We choose the following algorithm as the baseline algorithm:

NonFed: supervised learning tasks will be executed on 20 devices, but the federation framework will not be deployed on these clients.

FedAvg: an FL algorithm with SGD is executed. Additionally, the aggregation of global models uses a weighted average algorithm.

4.4. Results and Discussion

First, compared with the NonFed and FedAvg on user shopping behavior data, we analyze the accuracy of FedTCM. As the FedTCM is an asynchronous framework, we cannot compare the three methods in the same round. Therefore, we must set a time basis. We utilize system time on the central server. Additionally, the time baseline is the time consumed by FedAvg running 220 rounds. When the FedAvg ran 220 rounds, it almost has been of convergence. Additionally, the FedTCM and NonFed will run with the same time slice. To simulate the real-world situation, we assume that the clients compute at different speeds. Hence, we assign each client a different training speed via a delay time. The delay time is controlled by parameters c_t. In the other words, the delay time parameter c_t will control the calculation time of the slowest device, and the calculation time of other devices is $t \in [1, c_t]$. FedTCM and FedAvg consider the effect of device computational speed. In NonFed, we do not consider the effect of device computation speed, as it will be executed on a single client. In our experiment, FedAvg runs the least number of rounds in fixed time, followed by FedTCM, and finally by NonFed.

Figure 5 shows the accuracy of 3 methods with different degrees of data heterogeneity when batch size $B = 3$ and $c_t = 2$. In Figure 5a, when $\alpha = 10$, the data are lower in heterogeneity, and the model is less affected. The accuracy of NonFed fluctuates slightly from start to finish, and FedTCM is 10.1% more accurate than NonFed. In contrast, the federated method performs better in the case of low heterogeneity, and their accuracy curves are smooth with almost no fluctuations. Additionally, FedTCM is more accurate

than FedAvg by 1.2%. In Figure 5b, the degree of heterogeneity of the data is increased, and the accuracy curves of the three methods fluctuate to different degrees, especially the fluctuation of NonFed is the most obvious. Hence, NonFed is more sensitive to the degree of heterogeneity of the data. In the federated method, FedAvg converges slower and with a slight decrease in accuracy. FedTCM is not affected by Non-IID in the convergence phase, although the accuracy fluctuation is more obvious in the initial training phase. Additionally, the accuracy of FedTCM is 10.3% and 1.1% higher than NonFed and FedAvg, respectively. In Figure 5c, the degree of data heterogeneity further is increased. Affected by the heterogeneous data, the accuracy of NonFed becomes significantly reduced. Federated methods are also significantly impacted: the accuracy of FedTCM is still 10.4% higher than NonFed. Comparing the two federated methods, the accuracy fluctuations of FedAvg are more obvious and slightly reduced accuracy. However, FedTCM performs better than FedAvg: the accuracy of FedTCM is 1.1% higher than FedAvg. Additionally, whatever the degree of heterogeneity is, FedTCM always can achieve the highest accuracy among the three methods.

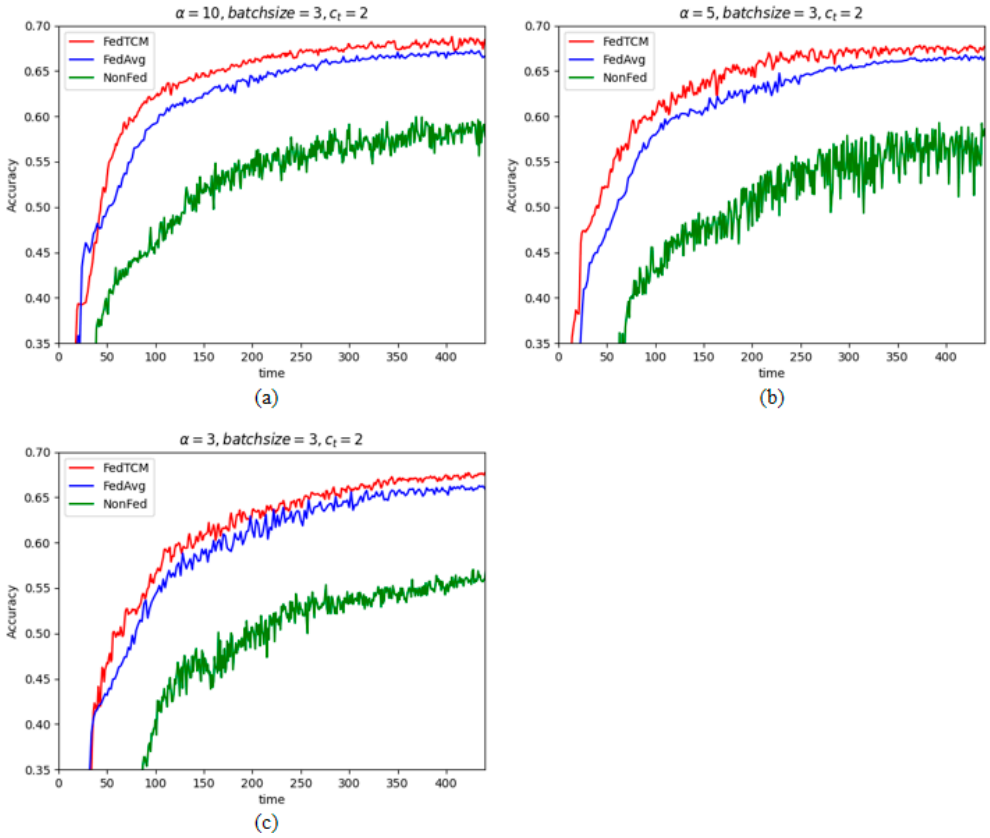

Figure 5. Comparing FedTCM with the baseline algorithms on the shopping behavior dataset at batch size. $B = 3, c_t = 2$ (**a**) FedTCM and baseline algorithms are all set $\alpha = 10$. (**b**) FedTCM and baseline algorithms are all set $\alpha = 5$. (**c**) FedTCM and baseline algorithms are all set $\alpha = 3$.

With changes in the degree of data heterogeneity, NonFed does not converge. The accuracy of FedTCM is on average 10.2% higher than NonFed. As the NonFed can only utilize the dataset of one client, the generalization ability of NonFed is the worst one. Thus, the federated methods mitigate the impact of Non-IID data better than NonFed. FedTCM and FedAvg use data from 20 devices under the federation framework. FedTCM converges in 350 system times, and FedAvg converges in 380 system times. Due to the simple parameter aggregation method of FedAvg, FedAvg is susceptible to the effects of non-IID data. Compared with FedAvg, FedTCM is more accurate and converges faster.

The experiments verified the effectiveness of the two-tier caching mechanism. FedTCM performs better than the baseline algorithm in the Non-IID shopping behavior data.

As FedTCM uses a two-tier cache mechanism and asynchronous communication between the server and cluster mediator, the server does not need to communicate with all clients but only with part of all cluster mediators. The server is not affected by the speed of client computing. Hence, FedTCM takes less time to run one round. Compared to FedAvg, FedTCM shortens the computation time of each round. Correspondingly, FedTCM ran more rounds than FedAvg in the specified time. As shown in Figure 5, for instance, if we use 65% as the target accuracy, FedTCM takes 157 system times to achieve the target accuracy, while FedAvg takes 228 system times when $\alpha = 10$. Additionally, when $\alpha = 5$, FedTCM takes 172 system time, and FedAvg takes 270 system time. When $\alpha = 3$, FedTCM requires 254 system time, and FedAvg requires 302 system time. Compared with FedAvg, FedTCM requires less time to achieve the target accuracy.

Since the user shopping behavior dataset is highly heterogeneous, three methods would be affected by different batch size B. The simulation results with batch size $B = 5$ and $c_t = 2$ are shown in Figure 6. In Figure 6a, compared to the results in Figure 5a, although NonFed can achieve higher accuracy, the accuracy fluctuates drastically. Additionally, the federated method has a slower convergence rate. The federated method is more stable than the NonFed, and the accuracy of FedTCM is 7.5% higher than NonFed. Moreover, the accuracy of FedTCM is 1% higher than that of FedAvg. Especially in Figure 6b, the degree of heterogeneity of the data increases, and the accuracy of NonFed is reduced significantly. Hence, NonFed is more sensitive to changes in parameters. At the same time, the accuracy of the federated methods decreased, especially the FedAvg accuracy decreased more. FedTCM is still the best performer among the three methods. The accuracy of FedTCM was 11.2% and 1% higher than NonFed and FedAvg, respectively. In Figure 6c, three methods are affected to different degrees as the data heterogeneity increases. The fluctuation of NonFed accuracy was severe. Although the accuracy of the federated method slightly decreases, the accuracy of the federated method is still higher than NonFed, especially FedTCM. FedTCM is 10.8% more accurate than NonFed. The federated method is affected drastically before 100 system time. Comparing the two federated methods, the accuracy of FedTCM is 1.2% higher than that of FedAvg.

The adjustment of batch size has an impact on all three methods. When increasing the value of batch size, the model will probably fall into local minima, which will affect the experimental results. In all experiments, as the degree of data heterogeneity increases, we note that NonFed still does not converge, and the accuracy of NonFed fluctuates drastically. NonFed is sensitive to parameter changes and has the worst generalization performance. FedTCM starts converging at 380 system times and FedAvg starts converging at 400 system times. Hence, the federated method still performs better than NonFed. Compared to Figure 5, although the convergence rate of the federated method is significantly lower, FedTCM performs better than FedAvg in both convergence speed and accuracy. FedTCM is most suited to process Non-IID data and it has excellent generalization capability.

Same as Figure 5, FedTCM takes less time to run one round. FedTCM ran more rounds than FedAvg in the specified time. In Figure 6, using 65% as the target accuracy, FedTCM takes 257 system time to achieve the target accuracy, while FedAvg takes 304 system time when $\alpha = 10$. When $\alpha = 5$, FedTCM takes 312 system time, and FedAvg takes 396 system time. When $\alpha = 3$, FedTCM requires 358 system time, and FedAvg requires 410 system time. Although we adjusted the experimental parameters, compared with FedAvg, FedTCM still requires less time to achieve the target accuracy.

To verify the influence of the client training speed on the experimental results, we design a group of experiments with different values of c_t. Figure 7 shows the comparison of FedTCM with the baseline method for different degrees of heterogeneity with $c_t = 4$. Since NonFed does not consider the effect of device computation speed between clients, we do not list the results of NonFed. As concluded in the previous section, even after considering the computational speed of the device, FedTCM outperforms FedAvg both in terms of

convergence speed and accuracy. The computational speed of the device does not affect the effectiveness of FedTCM.

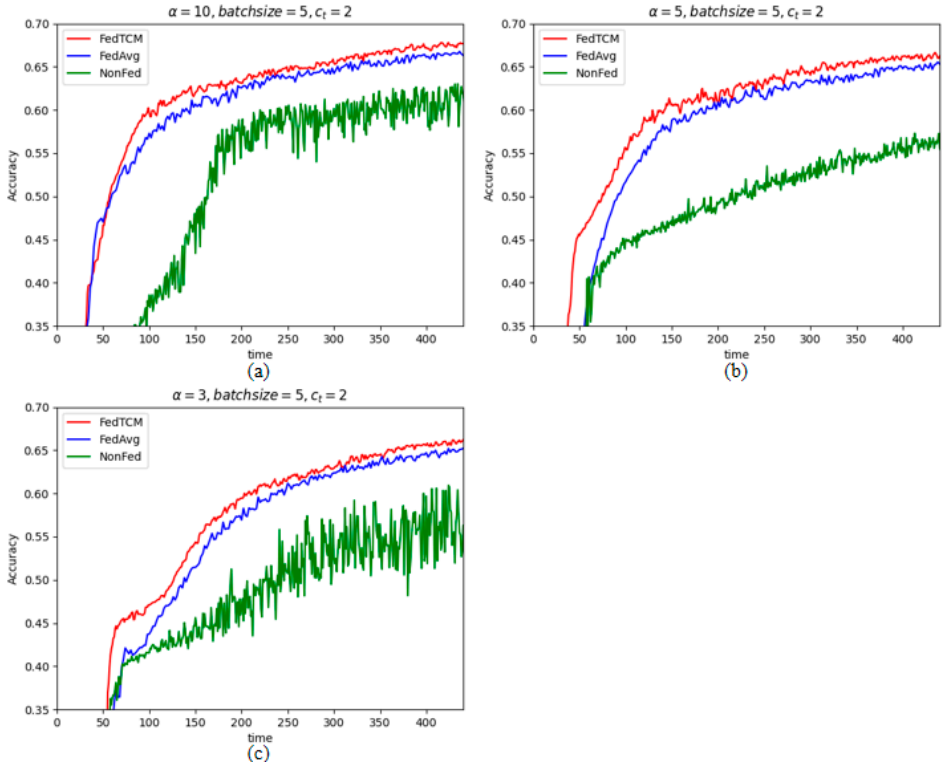

Figure 6. Comparing FedTCM with the baseline algorithm with different α at batch size $B = 5, c_t = 2$. (**a**) FedTCM and baseline algorithms are all set $\alpha = 10$. (**b**) FedTCM and baseline algorithms are all set $\alpha = 5$. (**c**) FedTCM and baseline algorithms are all set $\alpha = 3$.

In Figure 7 for the same reasons, FedTCM requires less time to achieve the target accuracy than FedAvg.

Figure 8 shows the accuracy of the 3 methods with different degrees of data heterogeneity when $c_t = 2$ in user sports behavior data. In Figure 8a, when $\alpha = 10$, the highest accuracy is achieved by all three methods. However, the accuracy curve of NonFed fluctuates drastically and does not converge. The accuracy of FedTCM is 14.4% higher than NonFed. In contrast, the federated methods can perform better with heterogeneous data, both in terms of accuracy and convergence speed. Compared to the typical federated algorithm FedAvg, the accuracy of FedTCM is 1.6% higher than it. In Figure 8b, as the degree of data heterogeneity rises, the accuracy of all three methods is affected, and the accuracy curve fluctuates more significantly, especially NonFed. The accuracy of FedTCM is 15.8% higher than NonFed. Additionally, in the federated method, despite the impact of heterogeneous data on FedTCM, the accuracy of FedTCM is 2.3% higher than FedAvg. In Figure 8c, the degree of data heterogeneity continues to rise, and the accuracy of all three methods is drastically affected. Nevertheless, the accuracy of FedTCM is still the highest among the three methods. The accuracy of FedTCM is 15.6% higher than NonFed and 2.2% higher than FedAvg.

In user sports behavior data, as the degree of data heterogeneity increases, NonFed has the largest drop in accuracy. Hence, NonFed is more sensitive to data changes. In Figure 8, the accuracy of the federated methods is much higher than NonFed, and the federated method is more suitable for heterogeneous data than NonFed. In contrast, the accuracy of FedTCM was, on average, 2.0% higher than FedAvg, and the fluctuations of the accuracy curve of FedTCM were the slightest. FedTCM performs better than the baseline algorithm in the Non-IID sports behavior data.

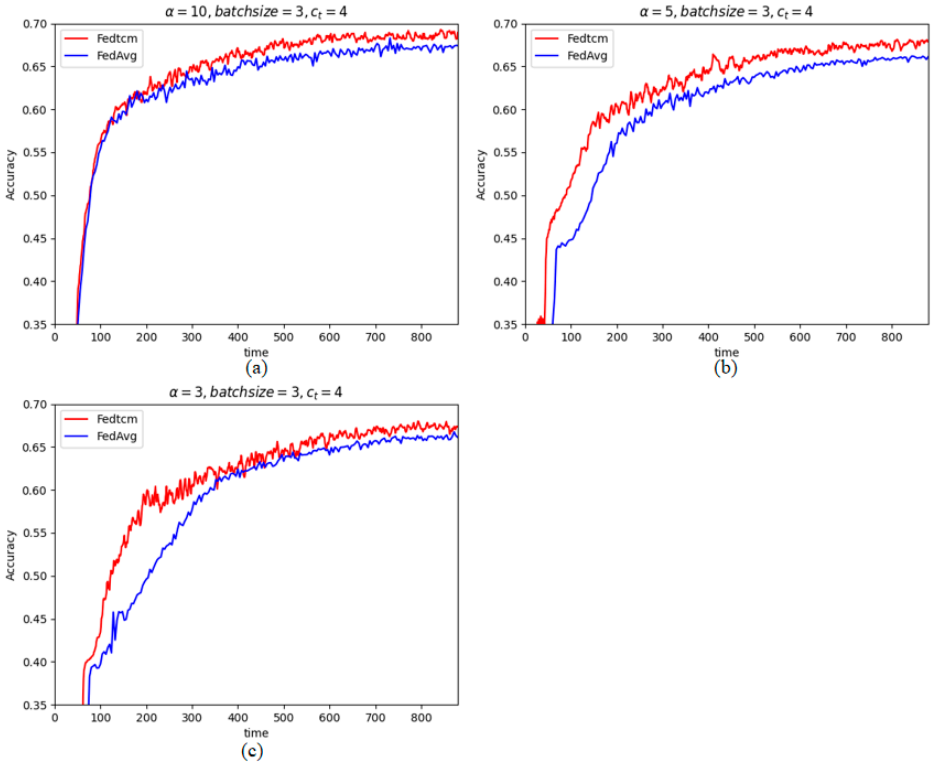

Figure 7. Comparing FedTCM with FedAvg with different α at batch size $B = 3, c_t = 4$. (**a**) FedTCM and FedAvg are all set $\alpha = 10$. (**b**) FedTCM and FedAvg are all set $\alpha = 5$. (**c**) FedTCM and FedAvg are all set $\alpha = 3$.

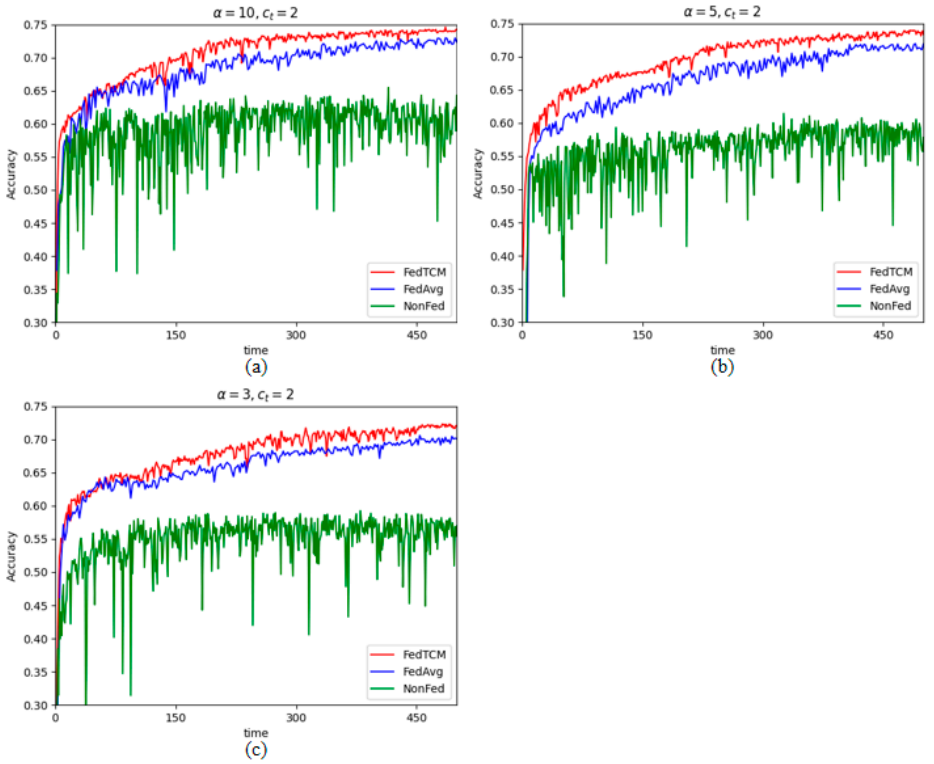

Figure 8. Comparing FedTCM with the baseline algorithms on the sports behavior dataset at $c_t = 2$. (**a**) FedTCM and baseline algorithms are all set $\alpha = 10$. (**b**) FedTCM and baseline algorithms are all set $\alpha = 5$. (**c**) FedTCM and baseline algorithms are all set $\alpha = 3$.

Additionally, the comparison of computational time between FedTCM and FedAvg, which is based on the sports behavior data, has a similar result with experiences on the shopping behavior data. FedTCM requires less time than FedAvg to achieve the same target accuracy.

Tables 2 and 3 show the number of model parameters sent/received by the client and server with different degrees of heterogeneity and different device computation speeds on user shopping behavior data. As the degree of heterogeneity increases, the number of model parameters sent/received by the server rises significantly. This occurs because changing the degree of heterogeneity of the data will affect the number of clusters. The number of clusters is highest in $\alpha = 3$ compared to $\alpha = 5$ and $\alpha = 10$. The more clusters there are, the less the cluster will be affected by slow computing clients and the more frequently the cluster will communicate with the server. Moreover, the number of servers sent/received in Table 2 is significantly less than in Table 3 because the computational speed of the device affects the overall experiment time. The slower the computation speed of the device, the longer the time it takes for the model to converge. The faster device does not wait for the slower device, so the faster device can communicate with the server more frequently. Therefore, when c_t increases, the number of servers sending/receiving increases. The number of models sent/received by local clients depends on the speed of computation of all clients in the cluster, Tables 2 and 3 show the average number of devices sent/received.

Table 2. Number of sent/received model parameters with different degrees of heterogeneity in FedTCM ($c_t = 2$), * denotes the result of averaging.

Device	α = 10	α = 5	α = 3
Central server	1981	2294	2521
Local client	260 *	287 *	256 *

Table 3. Number of sent/received model parameters with different degrees of heterogeneity in FedTCM ($c_t = 4$), * denotes the result of averaging.

Device	α = 10	α = 5	α = 3
Central server	2295	2690	3368
Local client	341 *	338 *	336 *

Table 4 illustrates the number of model parameters sent/received by FedTCM and FedAvg during a fixed experimental time on user shopping behavior data. The sent/received numbers of FedTCM in Table 4 are an average value of the sent/received numbers shown in Tables 2 and 3. Compared to FedAvg, although FedTCM has increased the number of sent/received on the local clients, it has significantly reduced the number of sent/received on the server. For local clients, the increase in the number of local training rounds is not catastrophically burdensome. However, for central servers in large network structures, reducing the communication burden can effectively reduce data congestion and increase the efficiency of communication and computation. Hence, the FedTCM can provide more excellent communication performance than FedAvg, too.

Table 4. Number of model parameters sent/received by different methods, * denotes the result of averaging.

Device	FedAvg	FedTCM (c_t = 2)	FedTCM (c_t = 4)
Central server	4400	2262 *	2784 *
Local client	220	268 *	338 *

From the client's perspective, FedTCM runs more rounds than FedAvg in the same amount of time. This demonstrates the effectiveness of our designed clustering mechanism

and asynchronous communication. Under the same conditions, FedTCM is more robust to device computation speed than FedAvg. As a result, FedTCM can fully utilize local computing resources and execute them more efficiently. From the server's perspective, FedTCM dramatically reduces the number of communications with the server. FedTCM differs from the typical FL algorithm which does not require all clients to communicate with the server. Since each cluster trains the model parameters independently, the aggregated model of the cluster mediator can represent the data information of all clients in the cluster. FedTCM can reduce the communication burden on the server while improving the accuracy of the model.

5. Conclusions

In this work, we proposed an FL algorithm, FedTCM, based on the two-tier cache mechanism. FedTCM can reduce the impact of Non-IID on user behavior modeling. Although the method without the federated framework can be trained, it cannot converge on the Non-IID data set. Compared to the method without the federated framework, FedTCM exhibits outstanding performance on Non-IID data. In the user shopping behavior dataset, the accuracy of FedTCM is 11.2% higher than NonFed at the maximum, 7.5% higher than NonFed at the minimum, and 10% higher than NonFed at the average under different degrees of data heterogeneity. In the user sports behavior dataset, the accuracy of FedTCM is 15.8% higher than NonFed at the maximum, 14.4% higher than NonFed at the minimum, and 15.2% higher than NonFed at the average for different degrees of data heterogeneity. Therefore, FedTCM has better generalization ability on Non-IID data. In the user shopping behavior dataset, the accuracy of FedTCM is 1.2% higher than FedAvg at maximum, 1% higher than FedAvg at minimum, and 1.1% higher than FedAvg at an average under different degrees of data heterogeneity. In the user sports behavior dataset, the accuracy of FedTCM is 2.3% higher than FedAvg at maximum, 1.6% higher than FedAvg at minimum, and 2% higher than FedAvg at an average under different degrees of data heterogeneity. Meanwhile, FedTCM converged faster than FedAvg, and FedTCM can provide more excellent communication performance than FedAvg. At the same time, in the convergence phase, the accuracy of the baseline algorithm is more volatile with decreasing the α, and in contrast, FedTCM maintained a smoother accuracy.

The goal of our proposed approach is to use a single model to mitigate the impact of Non-IID data on the model. A potential limitation is that a single model is an optimal solution for the global task, but with Non-IID data, it is not optimal for every client task. Even though the global model achieves the highest accuracy in the global data, it may not be suitable for each client because the local client data distribution is different from the global data distribution. In the future, we will consider federated learning approaches that generate multiple personalized models. Personalized models are created for each client through both the public knowledge from the other clusters and the specific knowledge of the current client, enhancing the generalizability of personalized models on different data distributions.

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

Funding: This paper is supported by the project "User Behavior Features Oriented Research on Analysis of Multi-Source Data in CDN" (20200401082GX), which is financially supported by the Science and Technology Development Program of Jilin Province, China.

Data Availability Statement: Due to the nature of this research, participants of this study did not agree for their data to be shared publicly, so supporting data is not available.

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

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 Conference, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
2. 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; Volume 2, pp. 429–450.
3. Gao, L.; Fu, H.; Li, L.; Chen, Y.; Xu, M.; Xu, C.-Z. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 10112–10121.
4. Karimireddy, S.P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; Suresh, A.T. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In Proceedings of the Machine Learning Research, 37th International Conference on Machine Learning, Vienna, Austria, 12–18 July 2020; pp. 5132–5143.
5. Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; Chandra, V. Federated learning with non-IID data. *arXiv* **2018**, arXiv:1806.00582.
6. Yao, X.; Huang, T.; Zhang, R.-X.; Li, R.; Sun, L. Federated learning with unbiased gradient aggregation and controllable meta updating. *arXiv* **2019**, arXiv:1910.08234.
7. Younis, R.; Fisichella, M. FLY-SMOTE: Re-balancing the non-IID iot edge devices data in federated learning system. *IEEE Access* **2022**, *10*, 65092–65102. [CrossRef]
8. Duan, M.; Liu, D.; Chen, X.; Tan, Y.; Ren, J.; Qiao, L.; Liang, L. Astraea: Self-balancing federated learning for improving classification accuracy of mobile deep learning applications. In Proceedings of the 2019 IEEE 37th International Conference on Computer Design (ICCD), Abu Dhabi, United Arab Emirates, 17–20 November 2019; pp. 246–254.
9. Xie, C.; Koyejo, S.; Gupta, I. Asynchronous federated optimization. *arXiv* **2019**, arXiv:1903.03934.
10. Hu, C.-H.; Chen, Z.; Larsson, E.G. Device scheduling and update aggregation policies for asynchronous federated learning. In Proceedings of the 2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Lucca Italy, 27–30 September 2021; pp. 281–285.
11. Jeong, E.; Oh, S.; Kim, H.; Park, J.; Bennis, M.; Kim, S.-L. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-IID private data. *arXiv* **2018**, arXiv:1811.11479.
12. Long, G.; Xie, M.; Shen, T.; Zhou, T.; Wang, X.; Jiang, J. Multi-center federated learning: Clients clustering for better personalization. *World Wide Web* **2023**, *26*, 481–500. [CrossRef]
13. Duan, M.; Liu, D.; Ji, X.; Liu, R.; Liang, L.; Chen, X.; Tan, Y. Fedgroup: Efficient federated learning via decomposed similarity-based clustering. In Proceedings of the 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), New York, NY, USA, 30 September–3 October 2021; pp. 228–237.
14. Huang, Y.; Chu, L.; Zhou, Z.; Wang, L.; Liu, J.; Pei, J.; Zhang, Y. Personalized cross-silo federated learning on non-IID data. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021; pp. 7865–7873.
15. Hsu, T.-M.H.; Qi, H.; Brown, M. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv* **2019**, arXiv:1909.06335.
16. Kopparapu, K.; Lin, E. Fedfmc: Sequential efficient federated learning on non-iid data. *arXiv* **2020**, arXiv:2006.10937.
17. Jamali-Rad, H.; Abdizadeh, M.; Singh, A. Federated learning with taskonomy for non-IID data. *IEEE Trans. Neural Netw Learn Syst.* **2022**, 1–12. [CrossRef] [PubMed]
18. Xue, Y.; Klabjan, D.; Luo, Y. Aggregation delayed federated learning. In Proceedings of the 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, 17–20 December 2022; pp. 85–94.
19. Khodak, M.; Balcan, M.-F.F.; Talwalkar, A.S. Adaptive gradient-based meta-learning methods. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 8–14 December 2019; pp. 5917–5928.
20. Shi, G.; Li, L.; Wang, J.; Chen, W.; Ye, K.; Xu, C. HySync: Hybrid federated learning with effective synchronization. In Proceedings of the 2020 IEEE 22nd International Conference on High Performance Computing and Communications (HPCC), Yanuca Island, Cuvu, Fiji, 14–16 December 2020; pp. 628–633.
21. Ghosh, A.; Hong, J.; Yin, D.; Ramchandran, K. Robust federated learning in a heterogeneous environment. *arXiv* **2019**, arXiv:1906.06629.
22. Ghosh, A.; Chung, J.; Yin, D.; Ramchandran, K. An efficient framework for clustered federated learning. *Adv. Neur. Inf. Process. Syst.* **2020**, *33*, 19586–19597. [CrossRef]