2022 IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Metaverse)

AFMeta: Asynchronous Federated Meta-learning with Temporally Weighted Aggregation

Sheng Liu[†], Haohao Qu[‡], Qiyang Chen[§], Weitao Jian[⋆], Rui Liu[‖], Linlin You[*¶]
[†‡§⋆¶]*School of Intelligent Systems Engineering, Sun Yat-Sen University, Shenzhen, China*
[‖]*School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore*
[†]liush235,[‡]quhaoh,[§]chenqy87@mail2.sysu.edu.cn, [⋆]jianwt5@mail2.sysu.edu.cn, [‖]rui.liu@ntu.edu.sg, [¶]youllin@mail.sysu.edu.cn

***Abstract*—The ever-increasing concerns on data security and user privacy have significantly impacted the current centralized mechanism of intelligent systems in bridging private data islands and idle computing resources commonly dispersed at the edge. To resolve that, a novel distributed learning paradigm, called Federated Learning (FL), which can learn a global model in a collaborative and privacy-preserving manner, has been proposed and widely discussed. Furthermore, to tackle the data heterogeneity and model adaptation issues faced by FL, meta-learning starts to be applied together with FL to rapidly train a global model with high generalization. However, since federated meta-learning is still in its infancy to collaborate with participants in synchronous mode, straggler and over-fitting issues may impede its application in ubiquitous intelligence, such as smart health and intelligent transportation. Motivated by this, this paper proposes a novel asynchronous federated meta-learning mechanism, called AFMeta, that can measure the staleness of local models to enhance model aggregation. To the best of our knowledge, AFMeta is the first work studying the asynchronous mode in federated meta-learning. We evaluate AFMeta against state-of-the-art baselines on classification and regression tasks. The results show that it boosts the model performance by 44.23% and reduces the learning time by 86.35%.**

***Index Terms*—Federated Learning, Federated Meta-learning, Asynchronous Federated Meta-learning, Model Aggregation**

I. Introduction

In recent years, the boom of Artificial Intelligence (AI), especially machine learning, has greatly benefited human beings to renovate services in various fields, e.g., smart cities [1], personal healthcare [2], autonomous transportation [3], etc. Since substantial and available data is a prerequisite for learning high-performance intelligent models, many AI-based systems rely on centralized data storage and model training. However, with the increasing attention on data security and user privacy, related regulations become ever more rigorous, making it infeasible and costly to collect and fuse dispersive and sensitive data into a unified dataset to implement centralized training as nowadays widely utilized.

To bridge the data islands caused by data security and user privacy, a novel distributed learning paradigm, called Federated Learning (FL) [4], has been proposed and widely discussed in academic and industrial communities. In general, a standard FL system consists of a server and multiple clients, in which, each client trains a local model based on its own data and uploads the model parameters to the server for global model aggregation. Then the server distributes the updated global update to each client to conduct a new round of local training. Since privacy-sensitive data are not exposed and scattered computing resources of clients are utilized, FL has been applied to many domains, such as mobility behavior analysis in autonomous transportation systems [5], and remote health monitoring in smart home [6].

However, since data collected separately by clients are related to their usage habits and local environments without information sharing, FL has an intrinsic issue of data heterogeneity, such as imbalanced data sizes, missing certain classes, and non-identical data distributions, impeding its further applications [7]. If such an issue can not be properly addressed, the model training process can be unstable and cost-inefficient, and learned models can be biased with limited performance [8]. Hence, as one of the solutions, meta-learning is adopted into FL [9], [10], named federated meta-learning (FMeta), to rapidly train an initial model with high generalization to the local task or data in each communication round.

Even though FMeta has been utilized in several real-world tasks to harness sensitive and heterogeneous data, e.g., parking occupancy prediction [11], [12], and wireless traffic prediction [13], it is still in its infancy, suffering straggler and over-fitting issues caused by the synchronous mode [9], [10]. In general, since the synchronous FMeta (SFMeta) requires all the clients to work under the same pave/schedule, stragglers with lower computing powers than other clients may affect the overall learning performance Besides, dropped parameters by stragglers will cause over-fitting on local models uploaded by clients with high success rate. Moreover, such issues encountered by SFMeta can be enlarged in ubiquitous systems, as clients at the edge have difficulty maintaining stable communication with the server due to resource constraints, leading to long learning time and high failure rate [8].

To tackle the issues in SFMeta, this paper incorporates the asynchronous mode into FMeta, called AFMeta, and proposes a temporally weighted mechanism for AFMeta. As shown in Figure 1, compared to SFMeta, the proposed AFMeta mechanism can ease the collaboration among clients with unblocked learning rounds to consistently receive local

* Corresponding author: Linlin You, e-mail: youllin@mail.sysu.edu.cn

979-8-3503-4655-8/22/$31.00 ©2022 IEEE
DOI 10.1109/SmartWorld-UIC-ATC-ScalCom-DigitalTwin-PriComp-Metaverse56740.2022.00100

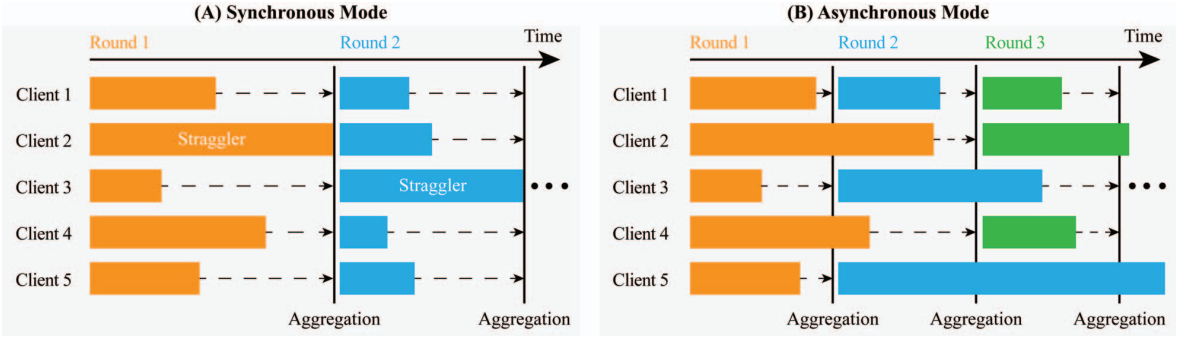

Fig. 1. The comparison between (A) synchronous mode and (B) asynchronous mode of federated meta-learning.

models from clients and update the global model once predefined conditions are satisfied, i.e., the maximum waiting time is reached. Such that, the influence of stragglers can be limited. Moreover, since the asynchronous mode may face temporal heterogeneity in local updates, i.e., the created time and received time of a model may be different, and the stale model can cause performance deterioration [14], AFMeta further implements a temporally weighted aggregation method to remedy the impact of over-fitting on stale models.

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

- To the best of our knowledge, this paper is the first in its kind to study AFMeta to support ubiquitous intelligence with straggler and over-fitting issues addressed;
- A temporally weighted aggregation method is proposed and applied in AFMeta, called AFMeta-TW, to tackle the temporal heterogeneity issue efficiently and effectively;
- Through a holistic evaluation based on three datasets, the proposed method can, on average, improve model performance by 44.23% and shorten learning time by 86.35%, while comparing it with state-of-the-art baselines in supporting both regression and classification tasks.

The remainder of this paper is organized as follows. Section II summarizes related work about asynchronous FL and FMeta. AFMeta is presented and evaluated in Section III and Section IV, respectively. Section V concludes the work and sketches the future research directions.

II. Related Work

This section discusses related work about asynchronous FL and FMeta.

A. Asynchronous Federated Learning (AFL)

As the first AFL algorithm, FedAsync [15] can update the global model immediately once the server receives a local model. Moreover, FedAsync introduces a mixture model aggregation method, in which, the updated global model is a mixture of the previous global model and the received local model. As the successor, ASO-Fed [16] inherits such an asynchronous manner to support online learning with local data accumulated during the training process.

However, such a mixture aggregation method may make the asynchronous learning process unsteady, and in turn, costly, as more interactions between the server and clients are required for the model to converge. Such that, a semi-asynchronous FL mechanism, called FedSA [17], is proposed, which determines the number of participants to be aggregated based on the communication budget to achieve time efficiency. Similar to FedSA, CSAFL [18] also optimizes the asynchronous mode by a spectral clustering algorithm dividing clients into groups according to the similarity of gradient direction and model latency to improve the overall learning performance.

Despite the timing to update the global model, it is also critical for AFL to harness the temporal difference among received local models [19], [20]. To resolve that, Chen et al. [21] aggregate deep layers and shallow layers of deep neural networks (DNNs) separately with different frequencies and assign larger aggregation weights to more recent models. Liu et al. [14] further explore these ideas by studying the best frequency of aggregation. Zhang et al. [22] introduce an asynchronous grouped federated learning framework (PAGFL) for IoT to update the global model based on the accuracy of each local model on a public validation dataset.

B. Federated Meta-learning (FMeta)

Meta-learning, or learning to learn, can encode meta-knowledge into an initialized model with high generalization [23]. As for the meta-model training, meta-learning algorithms, such as MAML (Mode-Agnostic-Meta-Learning) [24], [25] and its various variants, e.g., Reptile [26], have been proposed and widely used for gradient-based models in classification, regression, and reinforcement learning. Moreover, thanks to the generability of the meta-model, the over-fitting can be efficiently tackled and model localization or personalization can be easily conducted to support similar tasks in new contexts with optimized performance.

While comparing the learning procedure of FL and meta-learning, several similarities can be observed in local data processing and model updating steps [27]. Moreover, Jiang et al. [28] also demonstrate that FL and meta-learning have

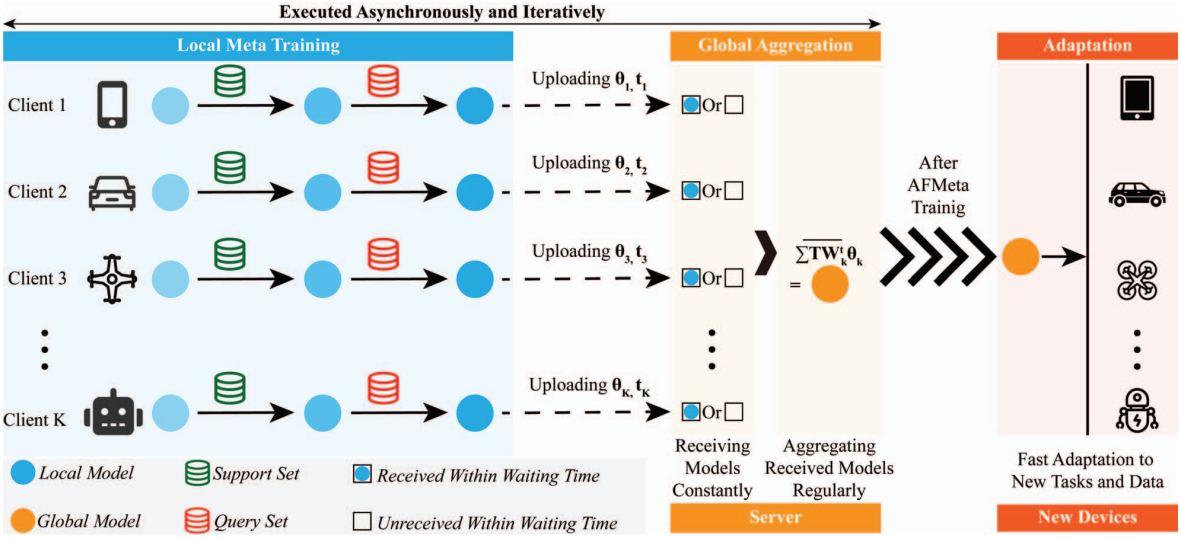

Fig. 2. The overview of AFMeta. The local meta training at the client side and the global aggregation at the server side are executed asynchronously and iteratively. After the above AFMeta training, the learned model can adapt to new tasks and data quickly with high performance.

mathematical connections. Hence, the integration of FL and Meta-learning becomes unencumbered [29], and currently, most studies focus on personalization [30]. However, these mechanisms may make the model converge slowly in edge computing. Such that, Yue et al. [31] devise a non-uniform client selection algorithm along with a resource allocation scheme to accelerate the learning speed.

Even though AFL and FMeta have been discussed, their integration in terms of asynchronous federated meta-learning (AFMeta) is still missing to tackle straggler [9], over-fitting [10], and data heterogeneity [7] issues in current solutions with FMeta in synchronous mode. To fill the gap, this paper introduces AFMeta and proposes a temporally weighted aggregation strategy to address the above issues efficiently and effectively.

III. Methodology

In this section, first, the problem definition is given; second, the local training process and the global aggregation process of AFMeta are introduced, respectively; third, the proposed temporally weighted method (TW) for AFMeta is illustrated; finally, the overall AFMeta with TW algorithm is summarized.

A. Problem Definition

As shown in Figure 2, we consider an asynchronous federated meta-learning system with K clients and one server. Specifically, each client has private local dataset D_k with labels, i.e., $D_k = (X_k, Y_k)$. Moreover, D_k is divided into support set D_k^{spt} and query set D_k^{qry} for meta-learning. Assume that each client has the abilities of model training and model uploading, however, the communication abilities are unstable due to resource constraints, i.e., the local model transmission time per client varies in each learning round.

Based on the above settings, the goals of this system are 1) to train an initialized model by managing heterogeneous clients asynchronously with low training cost (i.e., less learning time), and then, 2) to adopt the meta-model in classification or regression tasks rapidly with high model performance (i.e., high accuracy or low error). Accordingly, the optimization problem is defined in Formula 1,

$$\min_{\theta} \sum_{k=1}^{K} \mathcal{L}_k(\theta, D_k^{qry}) \;\&\; \min_{C} \sum_{t=1}^{\widehat{T}} c_i \tag{1}$$

where θ stands for the trained model; $\mathcal{L}_k$ is the loss function of client k, e.g., cross-entropy for classification tasks or mean square error for regression tasks; C is the accumulated training cost; $\widehat{T}$ is the first round when the desired performance target is reached; and c_i is the cost in the t^{th} round.

In summary, this paper intends to solve a dilemma between training cost and model performance, which commonly exists in current studies. Accordingly, a novel mechanism, called AFMeta with TW, is proposed to have both training time reduced and model performance improved.

B. Local Meta Training

A client k first receives the newest global model θ^{t_k} from the server, and clones it as the local model θ_k. Then, the updated model θ' is computed based on D_k^{spt} according to Formula 2,

$$\theta' = \theta_k - \alpha \nabla_{\theta_k} \mathcal{L}_k(\theta_k, D_k^{spt}) \tag{2}$$

where α is the inner learning rate.

We do not care about the performance of θ_k on D_k^{spt}, instead, the fast adaptability of θ_k is the objective. Thus, we

calculate the loss $\mathcal{L}(\theta', D_k^{qry})$ and use it to update local model according to Formula 3,

$$\theta_k = \theta_k - \beta \nabla_\theta \mathcal{L}(\theta', D_k^{qry}) \tag{3}$$

where β is the outer learning rate. In this way, the trained model θ_k is sensitive to local data changes and can adapt to new tasks quickly. However, the second-order derivation is required in Formula 3. To further reduce computation complexity, we adapt the first-order Model-Agnostic Meta-Learning [24] and rewrite Formula 3 into an approximate Formula 4.

$$\theta_k = \theta_k - \beta \nabla_{\theta'} \mathcal{L}(\theta', D_k^{qry}) \tag{4}$$

When the training is completed, each client k uploads the updated model θ_k to the server and waits for the new global model transmitted by the server to start the next local training.

C. Global Aggregation

In the designed asynchronous mode, the server receives local models from clients constantly and will execute the aggregation process when the pre-defined waiting time is reached. Note that the number of models participating in a round will change, since the communication capacities of clients vary over time, and in turn, the number of local models received by the server may be different per round. Such that, the aggregation of received local models in the current round to updating the global model can be conducted according to Formula 5,

$$\theta^{t+1} = \sum_{k=1}^{K_t} (\gamma_k \times \theta_k) \tag{5}$$

where K_t is the number of local models received in the t^{th} round, γ_k is the aggregation weight of client k, and θ^{t+1} is the updated global model. Note that in canonical FMeta, $\gamma_k = \frac{1}{K_t}$, indicating an average operation.

After the global model is updated, the server will distribute it to the clients to start local meta training if the stop condition is not matched (i.e., maximum round number); otherwise, ignite the model localization for model deployment.

D. Temporally Weighted Aggregation Strategy

As illustrated in Figure 1 (B), in asynchronous mode, some uploaded local models may be stale in the aggregation procedure, e.g., the local model of client 3 created in round 2 is to be aggregated in round 3. Intuitively, the latest model parameters, e.g., the model of client 1 created in round 3 deserve a higher aggregation weight than the stale local model mentioned above, as it provides the newest information. However, the average aggregation pattern in SFMeta may become inefficient to address such a temporal heterogeneity in asynchronous mode, as it considers local models equally.

Hence, in our study, a temporal weight (TW) is first proposed as defined in Formula 6,

$$TW_k^t = e^{-(t-t_k)} \tag{6}$$

where TW_k^t is the TW of client k in the t^{th} round, e is the natural logarithm, and t_k stands for the round when the local model of client k is created.

Then, TW is applied in the global aggregation, and accordingly, it transfers Formula 5 into Formula 7,

$$\theta^{t+1} = \sum_{k=1}^{K_t} (\overline{TW_k^t} \times \theta_k) \tag{7}$$

where $\overline{TW_k^t} = TW_k^t / \sum_{k=1}^{K_t} TW_k^t$ is the normalized weight. Such that, the more recent the local model is, the higher temporal weight it has in the aggregation.

As shown in Figure 3 (corresponding to the aggregation process of round 2 in Figure 1 (B)), by computing aggregation weights TW_k^t adaptively, θ^{t+1} can alleviate the effect of stale models and as a result, get to an optimal point compared with the average aggregation.

E. Algorithm of AFMeta with TW (AFMeta-TW)

As shown in Algorithm 1, AFMeta consists of two parts, namely:

- **Part 1: in each AFMeta Client.** First, AFMeta client k receives the global model θ_{t_k} from the server in the t_k round. Second, client k clones θ_{t_k} as its local model θ_k. Third, θ_k is updated according to Formulas 2 and 4. Finally, after the local meta training, θ_k and t_k are uploaded to the server for global aggregation.
- **Part 2: In the AFMeta Server.** First, the AFMeta server receives local models from clients continuously. Second, once the predefined waiting time is reached, the server will start the global aggregation, in which, normalized aggregation weights TW_k^t will be first calculated and applied in the aggregation function to update the global model. Finally, the updated global model θ_{t+1} is transmitted to clients and the next global round begins.

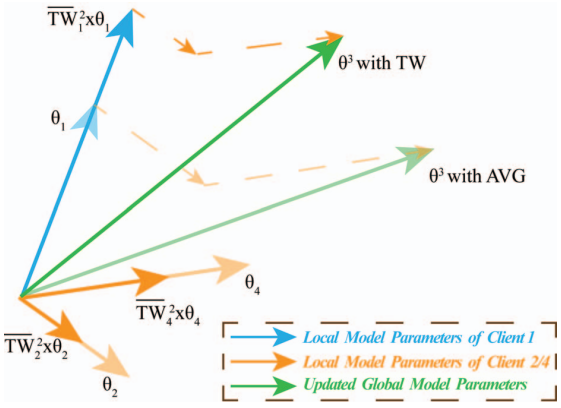

Fig. 3. The schematic diagram of temporally weighted aggregation strategy. Corresponding to the aggregation process of round 2 in Figure 1 (B), θ_1 is the newest model (blue line) while θ_2 and θ_4 are latent models (orange lines). The temporally weighted aggregation result (dark green line) can alleviate the effect of laten models compared with the average aggregation (light green line).

In summary, AFMeta implements unblocked local meta training and global aggregation processes to remedy the straggler and over-fitting issues. Moreover, a temporally weighted aggregation method is proposed to tackle the temporal heterogeneity of local models. Through AFMeta-TW, the overall learning performance can be improved significantly as revealed in Section IV-B.

Algorithm 1 AFMeta-TW

PART 1: Executed in each AFMeta client

for $k \leq K$ in parallel **do**
  Receiving the global model θ_{t_k}
  θ_k clones from θ_{t_k}
  Training local model according to Formulas 2 and 4
  uploading θ_k and t_k to the server
end for

PART 2: Executed in the AFMeta server

Thread 1: Receiving θ_k and t_k from clients consistently
Thread 2: Aggregating local models regularly
while the waiting time is reached **do**
  $TW^t = 0$
  for $k \leq K_t$ **do**
    calculating TW_k^t according to Formula 6
    $TW^t += TW_k^t$
  end for
  $\overline{TW_k^t} \leftarrow TW_k^t / TW^t$
  Aggregating according to Formula 7
  Transmitting θ_{t+1} to clients
  Increasing the global round t to $t+1$
end while

IV. Evaluation

This section first describes the experiment setting and then analyses the results of AFMeta. Finally, a further discussion based on the observations from the results is presented.

A. Experiment setting

The configuration consists of datasets and models, compared methods, and evaluation metrics, which will be described in the following sections, respectively.

1) Datasets and Models: Three datasets are utilized in the experiment, namely:

- **FMNIST:** Fashion-MNIST[1] (FMNIST) is a standard image classification dataset consisting of 60,000 training samples and 10,000 testing samples. Each image has the size of $28 \times 28 \times 1$ with a label from 10 classes (e.g., coat, sandal, etc.). According to a Non-IID setting introduced in FedAvg [4], the training and testing data are divided into 50 parts and 10 parts (each part contains 500-2,000 images) respectively. Moreover, to support meta-learning, each part is assigned to a training or testing client to train a CNN (Convolutional Neural Network), which has two convolution layers (16 and 32 channels, respectively) followed by a 2×2 max-pooling layer, two fully connected layers (128 and 256 units, respectively), and an output layer.
- **CIFAR-10:** CIFAR-10[2] is also a standard dataset that compromises a training set of 50,000 images, a testing set of 10,000 images, and 10 classes. Different from FMNIST, each sample is a $32 \times 32 \times 3$ color image. The data partition setting and CNN model structure are the same as the ones of FMNIST.
- **CHARGE:** The charge pile occupancy dataset[3] (which is crawled from the web by our team) contains 33 charging station data in Shenzhen, China, from 2021.12.09 to 2022.01.07, and the resolution is 5 minutes. Each station is treated as a client, and the task is to predict its charge pile occupancy. Furthermore, 28 of 33 stations are used for training, and the rest are used for testing. Finally, an RNN (Recurrent Neural Network) is designed to be trained with a single GRU (Gated Recurrent Unit).

As for the communication capability, the transmission time of each client ranges from 5s to 35s randomly, and the predefined waiting time for the server to start the global aggregation is 8s (besides the initial round will take 20s to have more local models aggregated). In summary, the default training settings are summarized in Table I. Note that the training clients are orchestrated during the training for the global model, and the testing clients are untouched during the training and only used for evaluation.

TABLE I
THE SUMMARY OF TRAINING SETTINGS.

Term	FMNIST	CIFAR-10	CHARGE
Local epoch	1	2	1
Inner learning rate	0.001	0.001	0.003
Outer learning rate	0.001	0.001	0.003
Batch size	48	48	3
Support:Query sets	3:2	3:2	3:2

2) Compared Methods: To better evaluate AFMeta and AFMeta equipped with the proposed weighted aggregation strategy, the following four methods are compared:

- **FedAvg:** The most recognized synchronous federated learning method [4];
- **SFMeta-aver:** The synchronous federated meta-learning method with average aggregation [9];
- **AFMeta-aver:** The proposed AFMeta mechanism with average aggregation;

[1]https://github.com/zalandoresearch/fashion-mnist

[2]http://www.cs.toronto.edu/ kriz/cifar.html

[3]https://github.com/nobody910/charge-station-dataset

TABLE II
THE SUMMARIZED RESULTS OF BASELINES AND AFMETA

Method	FMNIST			CIFAR-10			CHARGE				
	Accuracy↑	Loss↓	Time↓ (s)	Accuracy↑	Loss↓	Time↓ (s)	Loss↓	MAE↓	R^2 ↑	RMSE↓	Time↓ (s)
FedAvg	82.17%	0.4900	-	51.86%	1.3668	6,479	0.0507	0.1907	-0.0611	0.2235	-
SFMeta-aver	84.97%	0.4144	-	54.36%	1.3232	4,890	0.0153	0.1049	0.5434	0.1219	11,311
AFMeta-aver	87.15%†	0.3742	4,212	67.26%	0.9528	1,084	0.0033	0.0362	0.8798	0.0550	1,820
AFMeta-TW	**88.77%***	**0.3164**	**3,468**	**69.41%**	**0.8915**	**1,004**	**0.0032**	**0.0332**	**0.8830**	**0.0541**	**756**
Improv.1‡	4.47%	23.65%	-	27.49%	32.63%	79.47%	79.08%	68.35%	62.53%	55.62%	93.32%
Improv.2§	1.86%	15.45%	17.66%	3.20%	6.43%	7.38%	3.03%	8.29%	0.36%	1.64%	58.46%

* Bold numbers are the best performance.
† Numbers with underlines are the second best values.
‡ Improv.1 shows the percentage improvement of AFMeta-TW over SFMeta-aver.
§ Improv.2 shows the percentage improvement of AFMeta-TW over AFMeta-aver.

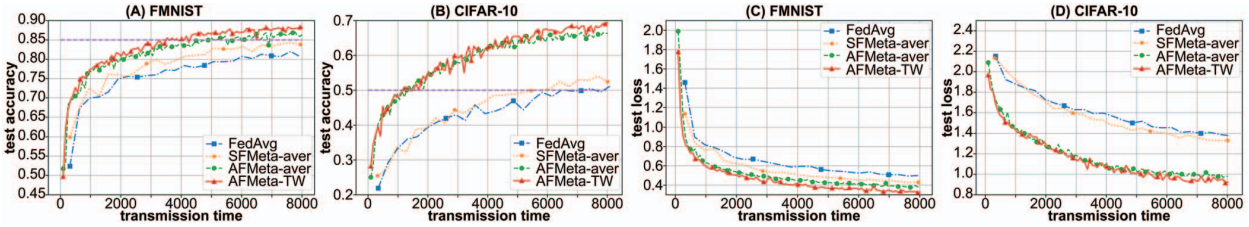

Fig. 4. The accuracy curves in (A) FMNIST; and (B) CIFAR-10, and the loss curves in (C) FMNIST; and (D) CIFAR-10.

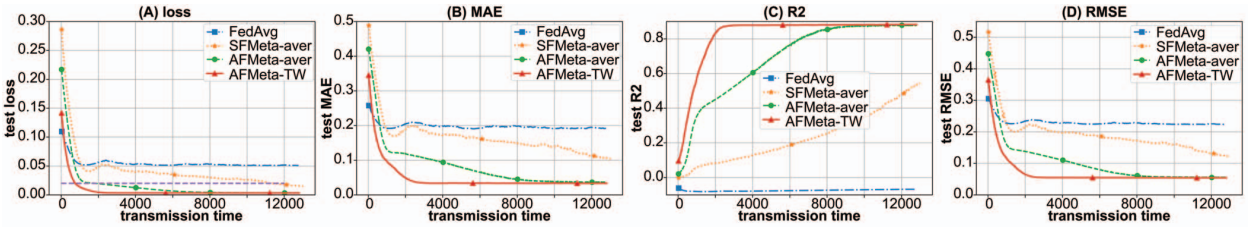

Fig. 5. The learning curves in CHARGE of (A) loss; (B)MAE; (C) R2; and (D) RMSE.

- **AFMeta-TW:** The proposed AFMeta mechanism with temporally weighted aggregation strategy.

By comparing FedAvg with FMeta methods (i.e., SFMeta-aver, AFMeta-aver, and AFMeta-TW), the effects of integrating meta-learning with FL can be revealed. Moreover, by comparing two AFMeta methods with SFMeta-aver, the pros and cons of FMeta by using synchronous and asynchronous modes can be analyzed and discussed.

3) Evaluation Metrics: As for the classification tasks performed on FMNIST and CIFAR-10 datasets, three metrics are utilized, namely:

- **Accuracy:** The most commonly used measurement;
- **Loss:** Cross entropy loss is used, which is the most recognized loss for classification tasks;
- **Training speed:** The time when the target accuracy is reached. Note that the target accuracies for FMNIST and CIFAR-10 are 85% and 50%, respectively.

Specifically, the model accuracy and loss are calculated according to Formula 8,

$$\begin{aligned} accuracy &= \frac{TP+TN}{TP+FP+FN+TN} \\ loss_{CE} &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{M} y_{ic} log(p_{ic}) \end{aligned} \tag{8}$$

where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively; N and M is the number of samples and categories, respectively; $y_{ic} = 1$ if sample i belongs to category c; otherwise, $y_{ic} = 0$; and p_{ic} is the predicted probability;

As for the regression task in CHARGE dataset, five metrics are used, namely:

- **Loss:** Mean Square Error, which is the loss commonly used for regression tasks;
- **MAE:** Mean Absolute Error;
- $\mathbf{R^2}$: Coefficient of Determination;
- **RMSE:** Root Mean Square Error.

- **Training speed:** The time when the target loss is reached. Note that the target loss for CHARGE is 0.02.

These metrics are computed according to Formula 9,

$$\begin{aligned} loss_{MSE} &= \frac{1}{N}\sum_{i=1}^{N}(\widehat{y_i}-y_i)^2 \\ MAE &= \frac{1}{N}\sum_{i=1}^{N}|\widehat{y_i}-y_i| \\ R^2 &= 1-\frac{\sum_{i=1}^{N}(\widehat{y_i}-y_i)^2}{\sum_{i=1}^{N}(y_i-\bar{y_i})^2} \\ RMSE &= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\widehat{y_i}-y_i)^2} \end{aligned} \tag{9}$$

where y_i and $\widehat{y_i}$ represent the actual value and the predicted value, respectively.

B. Performance Evaluation

Table II reports the performance comparing AFMeta with the baseline methods. From the results, three observations can be obtained, namely:

- **The three federated meta-learning methods outperform FedAvg significantly.** FedAvg performs the worst in the three datasets while the three federated meta-leaning methods have satisfactory performances for classification and regression tasks. It indicates the strengths of meta-learning in the fast adaptation.
- **AFMeta-aver superiors SFMeta-aver in model performance.** In FMNIST, AFMeta-aver can reach an accuracy of 87.15% and a loss of 0.3742, and hence, the corresponding improvements of 2.57% and 9.70% are observed compared with SFMeta-aver. Moreover, the accuracy curve and loss curve of AFMeta-aver can always be above SFMeta-aver as shown in Figure 4 (A) and Figure 4 (C). As for CIFAR-10, AFMeta-aver can surprisingly boost the accuracy and reduce the loss by about 23.73% and 27.99%, respectively. Figure 4 (B) and Figure 4 (D) also prove such promotions. In the case of CHARGE, the improvements in loss, MAE, R^2, and RMSE are 78.43%, 65.49%, 61.91%, 54.88%, respectively, and the advantages of asynchronous mode are also reflected in Figure 5.
- **AFMeta-TW achieves the best results compared with other methods.** As summarized in the Improv.1 and Improv.2 of Table II, AFMeta-TW performs the best on all metrics of the three datasets with significant improvements against SFMeta-aver and AFMeta-aver. Specifically, AFMeta-TW can achieve the highest accuracies of 88.77% and 69.41% for FMNIST and CIFAR-10 respectively, and the lowest loss of 0.0032 for CHARGE. By averaging the values of Improv.1 in Table II, the average increase in model performance is 44.23% compared with SFMeta-aver. The learning curves of AFMeta-TW are also the greatest as illustrated in Figure 4 and Figure 5.

C. Evaluation of Time Efficiency

First, as illustrated in Table II, Figure 4 and Figure 5 (A), both FedAvg and SFMeta train models slowly, and even can not reach the target accuracy of FMNIST at the end of learning.

Second, the training speed of AFMeta-aver is much faster than the two synchronous modes. Specifically, AFMeta-aver can reach the target accuracy or loss within 4,212, 1,048, and 1,820 seconds for FMNIST, CIFAR-10, and CHARGE, respectively; and compared with SFMeta-aver, the accelerations are 77.83% and 83.91% for CIFAR-10 and CHARGE, respectively. Note that FMNIST is not compared, as SFMeta-aver can not reach its target accuracy.

Finally, AFMeta-TW is observed to have the fastest training speed with improvements of about 17.66%, 7.38%, and 58.46% against AFMeta-aver for FMNIST, CIFAR-10, and CHARGE, respectively. When compared with the baseline SFMeta-aver, the average boost (average values of Improv.1 in Table II) is 86.35%.

D. Discussion

First, as illustrated in Figure 4 and 5, there are visible differences in learning curves between AFMeta-TW and synchronous methods for both classification and regression tasks, which indicates three prominent advantages of AFMeta-TW in improving the model performance, shortening the transmission time, and maintaining a stable learning process.

Second, the three advantages indicate that asynchronous mode can not only learn a model more quickly than synchronous mode but also train a better model in terms of accuracy, loss, etc., thus being more suitable to support ubiquitous systems that require high-performance models fast.

Finally, compared with AFMeta-aver, AFMeta-TW performs better, which proves the efficiency and effectiveness of the temporally weighted aggregation strategy. Although for the more complicated task CIFAR-10, the curve of AFMeta-aver is similar to AFMeta-TW in the first half learning phase according to Figure 4 (B), in the second half phase, there still exists a distinct difference, which shows that temporally weighted aggregation strategy can, indeed, tackle stale models in various scenarios.

V. Conclusion

This paper proposes a novel asynchronous meta-learning mechanism, called AFMeta, in which, an unblocked aggregation process is implemented to resolve issues about stragglers and over-fitting commonly occurring in the synchronous mode of FMeta. Moreover, to tackle the temporal heterogeneity issue faced by the asynchronous mode, a temporally weighted aggregation strategy is introduced to update the global model adaptively according to the staleness of local models. To the best of our knowledge, as the first attempt to incorporate asynchronous mode with federated meta-learning, the proposed AFMeta can, on average, boost model performance by 44.23%, reduce training time by 86.35%, and maintain

a more steady learning process, compared to state-of-the-art baselines to support both classification and regression tasks. Such a result indicates that the asynchronous mode in federated meta-learning superiors the synchronous mode in overall model performance and training efficiency to support ubiquitous systems.

In the future, the proposed aggregation strategy based on TW can be further enhanced to learn adaptive weights automatically by using machine learning methods. Therefore, a reinforcement learning-based algorithm will be studied to compute the optimal weights for each client according to the awards measured by the overall learning improvement per round (i.e., the accuracy improvement). Moreover, it is still undiscussed how to reduce communication burdens in AFMeta clients to impel its actual applications in ubiquitous IoT (Internet of Things) systems. Such that, a hierarchical communication topology with layer-wise transmission methods will be explored for a cost-efficient AFMeta.

ACKNOWLEDGMENT

This research was funded by the National Natural Science Foundation of China (62002398) and the Collaborative Innovation Center for Transportation of Guangzhou (202206010056).

REFERENCES

[1] L. You, B. Tunçer, R. Zhu, H. Xing, and C. Yuen, "A synergetic orchestration of objects, data, and services to enable smart cities," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 496–10 507, 2019.

[2] F. Firouzi, B. Farahani, M. Daneshmand, K. Grise, J. Song, R. Saracco, L. L. Wang, K. Lo, P. Angelov, E. Soares, P.-S. Loh, Z. Talebpour, R. Moradi, M. Goodarzi, H. Ashraf, M. Talebpour, A. Talebpour, L. Romeo, R. Das, H. Heidari, D. Pasquale, J. Moody, C. Woods, E. S. Huang, P. Barnaghi, M. Sarrafzadeh, R. Li, K. L. Beck, O. Isayev, N. Sung, and A. Luo, "Harnessing the power of smart and connected health to tackle covid-19: Iot, ai, robotics, and blockchain for a better world," *IEEE Internet of Things Journal*, vol. 8, no. 16, pp. 12 826–12 846, 2021.

[3] L. You, J. He, W. Wang, and M. Cai, "Autonomous transportation systems and services enabled by the next-generation network," *IEEE Network*, vol. 36, no. 3, pp. 66–72, 2022.

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

[5] L. You, J. He, J. Zhao, and J. Xie, "A federated mixed logit model for personal mobility service in autonomous transportation systems," *Systems*, vol. 10, no. 4, 2022.

[6] D. C. Nguyen, Q.-V. Pham, P. N. Pathirana, M. Ding, A. Seneviratne, Z. Lin, O. Dobre, and W.-J. Hwang, "Federated learning for smart healthcare: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, pp. 1–37, 2022.

[7] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[8] L. You, S. Liu, Y. Chang, and C. Yuen, "A triple-step asynchronous federated learning mechanism for client activation, interaction optimization, and aggregation enhancement," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

[9] F. Chen, M. Luo, Z. Dong, Z. Li, and X. He, "Federated meta-learning with fast convergence and efficient communication," *arXiv preprint arXiv:1802.07876*, 2018.

[10] X. Li, Y. Li, J. Wang, C. Chen, L. Yang, and Z. Zheng, "Decentralized federated meta-learning framework for few-shot multitask learning," *International Journal of Intelligent Systems*, 2022.

[11] H. Qu, S. Liu, Z. Guo, L. You, and J. Li, "Improving parking occupancy prediction in poor data conditions through customization and learning to learn," in *International Conference on Knowledge Science, Engineering and Management*. Springer, 2022, pp. 159–172.

[12] H. Qu, S. Liu, J. Li, Y. Zhou, and R. Liu, "Adaptation and learning to learn (all): An integrated approach for small-sample parking occupancy prediction," *Mathematics*, vol. 10, no. 12, p. 2039, 2022.

[13] L. Zhang, C. Zhang, and B. Shihada, "Efficient wireless traffic prediction at the edge: A federated meta-learning approach," *IEEE Communications Letters*, vol. 26, no. 7, pp. 1573–1577, 2022.

[14] S. Liu, Q. Chen, and L. You, "Fed2a: Federated learning mechanism in asynchronous and adaptive modes," *Electronics*, vol. 11, no. 9, p. 1393, 2022.

[15] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

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

[17] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "Fedsa: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3654–3672, 2021.

[18] Y. Zhang, M. Duan, D. Liu, L. Li, A. Ren, X. Chen, Y. Tan, and C. Wang, "Csafl: A clustered semi-asynchronous federated learning framework," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–10.

[19] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, "Fedat: a high-performance and communication-efficient federated learning system with asynchronous tiers," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–16.

[20] C.-H. Hu, Z. Chen, and E. G. Larsson, "Device scheduling and update aggregation policies for asynchronous federated learning," in *2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2021, pp. 281–285.

[21] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 10, pp. 4229–4238, 2020.

[22] T. Zhang, A. Song, X. Dong, Y. Shen, and J. Ma, "Privacy-preserving asynchronous grouped federated learning for iot," *IEEE Internet of Things Journal*, vol. 9, no. 7, pp. 5511–5523, 2022.

[23] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 5149–5169, 2022.

[24] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[25] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine, "Online meta-learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1920–1930.

[26] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999*, 2018.

[27] S. Lin, L. Yang, Z. He, D. Fan, and J. Zhang, "Metagater: Fast learning of conditional channel gated networks via federated meta-learning," in *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. IEEE, 2021, pp. 164–172.

[28] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *arXiv preprint arXiv:1909.12488*, 2019.

[29] S. Lin, G. Yang, and J. Zhang, "A collaborative learning framework via federated meta-learning," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 289–299.

[30] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3557–3568, 2020.

[31] S. Yue, J. Ren, J. Xin, D. Zhang, Y. Zhang, and W. Zhuang, "Efficient federated meta-learning over multi-access wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 5, pp. 1556–1570, 2022.