MSSA-FL: High-Performance Multi-stage Semi-asynchronous Federated Learning with Non-IID Data

Xiaohui Wei, Mingkai Hou, Chenghao Ren(✉), Xiang Li(✉), and Hengshan Yue

College of Computer Science and Technology, Jilin University, Changchun, China
{weixh,lixiang_ccst}@jlu.edu.cn,
{houmk20,rench19,yuehs18}@mails.jlu.edu.cn

Abstract. Federated Learning (FL) is an emerging distributed machine learning framework that allows edge devices to collaborative train a shared global model without transmitting their sensitive data to centralized servers. However, it is extremely challenging to apply FL in practical scenarios because the statistics of the data across edge devices are usually not independent and identically distributed (Non-IID), which will introduce the bias to global model. To solve the above data heterogeneity issue, we propose a novel *Multi-Stage Semi-Asynchronous Federated Learning* (MSSA-FL) framework. MSSA-FL benefits convergence accuracy through making the local model complete multi-stage training within the group guided by combination module. To improve the training efficiency of the framework, MSSA-FL adopts a semi-asynchronous update method. Meanwhile, proposed model assignment strategy and model aggregation method further boost the performance of MSSA-FL. Experiments on several public datasets show that MSSA-FL achieves higher accuracy and faster convergence than the comparison algorithms.

Keywords: Federated learning · Non-IID · Multi-stage · Semi-asynchronous

1 Introduction

Due to the rapid advance and remarkable achievements of artificial intelligence technology [1,2], a growing number of complicated intelligent tasks are pushed to edge devices. How to use private data on edge devices to jointly train a sophisticated machine learning model [3–5] has drawn an unprecedented level of attention. As one of the most recognized approaches to edge training, *Federated Learning* (FL) expects to leverage multiple devices to collaboratively train a shared machine learning model with their local dataset without private data transmission. Through aggregating weight updates from each device, a global model can be efficiently and securely obtained.

This work is supported by the National Natural Science Foundation of China (NSFC) (Grants No. U19A2061), National key research and development program of China under Grants No. 2017YFC1502306.

G. Memmi et al. (Eds.): KSEM 2022, LNAI 13369, pp. 172–187, 2022.
https://doi.org/10.1007/978-3-031-10986-7_14

A serious problem arises when applying FL to practical training: the data among devices are heterogeneous, i.e., non-independent and identically distributed data (Non-IID) [6,7]. Numerous researches have demonstrated that when training on heterogeneous data, local models based on the same initial model eventually converge to different models, which introduces unpredictable bias to the global model and results the deterioration in accuracy and convergence speed. Previous studies are mainly devoted to addressing the problem of Non-IID data in FL from two main perspectives. In a part of the work, sophisticated federated optimization methods are explored to restrict the weight divergence of local models. For example, FedProx [8] tackled the data heterogeneity by adding a bound term on the locally trained optimization objective. For preventing unstable and slow convergence on Non-IID data, SCAFFOLD [9] used variance reduction to correct the "client-drift" in each client's local updates. Methods in another part of the work such as data extension are used to generate homogeneous data in FL. Zhao et al. proposed to use globally shared data for training to deal with Non-IID. Astrea [10] alleviated the global data imbalance based on Z-score data augmentation and solved the local imbalance by mediator-based multi-device rescheduling.

However, the above researches mainly focus on improving the convergence accuracy, neglecting the training efficiency problems [11–13] that arise when FL is applied to heterogeneous devices. Since it must wait for the straggler devices before aggregating in each round, the server aggregation timing is actually determined by the slowest straggler, which increasing the time per round. Furthermore, fast training devices are idle for a long period of time until aggregation even though they have the capacity to engage in continuous training, leading to low resource utilization.

To solve the above issues, we propose a high-performance *Multi-Stage Semi-Asynchronous FL* (MSSA-FL) framework. Inspired by one of our key insight that unbalanced data can be combined into a large data set approximately balanced, MSSA-FL adopts a group-based multi-stage training method. Especially, combination module aimed at clustering devices with similar data distribution and grouping clusters with complementary data distribution. Through flexible multi-stage training strategy, MSSA-FL obtains approximately IID dataset training effect within group. To achieve the goal of efficiency optimization under the accuracy requirement, our proposed semi-synchronous update method with reasonable aggregation conditions boosts the frequency of model updates and allows more idle devices to participate in the training. To mitigate the impact of model staleness, we propose a heuristic aggregation method by adjusting the model aggregated weights. Considering the impact of different devices on the performance, in our model assignment strategy, we mainly exploring the training order of devices which include the hard-to-train labels and strategically increasing the participation opportunities for capable facilities to obtain the further performance improvement. In summary, the main contributions of our work are as outlined as follows:

- We propose a novel *multi-stage semi-asynchronous federated learning* (MSSA-FL) framework to solve the heterogeneous data issue and improve the training efficiency.
- We design a combination module for MSSA-FL to infer devices data distribution and group devices for the purpose of following multi-stage training. Guided by grouping results, we propose a multi-stage training method to make models trained on an approximate IID dataset combination, achieving the effect of improving the global model accuracy.
- We propose a semi-asynchronous update method to improve the low efficiency of synchronous FL. Furthermore, combining the characteristics of MSSA-FL, a model assignment strategy and an aggregation method for stale models are proposed to further boost the performance of MSSA-FL.
- We conduct extensive experiments to evaluate the effectiveness of the MSSA-FL algorithm on different publicly available datasets. The experimental results show that our approach is effective in both improving model accuracy and accelerating model training.

2 Related Work

The further integration of edge computing [14,15] and artificial intelligence [16,17] has promoted the rapid advancement of federated learning [2]. Federated learning (FL) [18], considered one of the most emerging distributed training techniques, provides a solution for privacy-preserving distributed machine learning. At present, multiple works are devoted to applying FL in real-world scenarios, which are mainly embodied in the following two aspects: (1) achieving higher convergence accuracy of FL with heterogeneous data [19,20,20]. (2) boosting the efficiency of FL.

Convergence Accuracy Improvement. Several efforts had been made to cope with the deterioration in accuracy and convergence speed of FL due to Non-IId data. Li et al. proposed FedProx [8] by adding a heterogeneity bound to tackle heterogeneity. Zhao et al. [7] used the *earth mover's distance* (EMD) to quantify data heterogeneity and proposed to use globally shared data for training to deal with Non-IID. FedMA [21] demonstrated that the result of aggregation can be affected by the permutations of layers and proposed a layer-wise aggregation strategy to improve global model accuracy. Duan et al. proposed Astrea [10] where devices with complementary data are divided into a group every round by mediators and the global model is obtained by synchronously aggregating all mediators' models which are trained sequentially on devices within groups.

Efficiency Improvement. In terms of boosting efficiency of FL, Nishio et al. proposed FedCS [22], in which server selects as many devices as possible to train the model in a limited time interval based on the resource of devices. Xie et al. proposed asynchronous FL [23], where the server performs aggregation as soon as it receives a local model from any device. Wu et al. proposed a new semi-asynchronous federated learning protocol SAFA [24] that classifies devices into

latest, stale and tolerable and used discriminative aggregation with caching to solve the problem of inefficient rounds and poor convergence speed of federated learning in extreme cases. In [25], Xu et al. proposed a "soft-training" approach to accelerate the stragglers by compressing the training models of stragglers.

However, there are few efforts to improve the overall performance of FL by considering convergence accuracy and efficiency together. We propose a high-performance *multi-stage semi-asynchronous FL* framework (MSSA-FL), considering the degradation of model accuracy due to Non-IID data and the inefficiency of synchronous updates on heterogeneous devices.

3 MSSA-FL: Multi-stage Semi-asynchronous Federated Learning

3.1 Framework Overview

The overall framework of MSSA-FL is shown in Fig. 1. High-performance multi-stage semi-asynchronous federated learning contains the following components: Multi-Stage Training, Semi-Asynchronous Training, Model Assignment and Model Aggregation. It is worth noting that the combination module in server as the key part of the framework solve the problem of data heterogeneity through clustering and grouping. Devices with similar data distribution are divided into clusters, and multiple clusters with complementary data distribution form a group.

The detail workflow of MSSA-FL can be described as follows. *Step 1:* Server starts one round pre-training and divides the devices into clusters and groups through the combination module. *Step 2:* Selected devices download the global model. *Step 3:* Devices train the received model on local dataset and upload the weight updates to the server after completion. *Step 4:* Server receives the local models and judges whether they have completed the multi-stage training. The completed models are aggregated to update the global model. While the unfinished models are sent to the corresponding devices according to their missing training stages and repeats *Step 2, 3*.

We will introduce our proposed methods in detail in the following subsections.

3.2 Combination Module and Multi-stage Training

Attributing the effect of data heterogeneity to the fact that local models trained on data with different data distributions have different optimization directions and unpredictable biases would be introduced into the global model through model aggregation. Therefore, in this section, we design a combination module to achieve an approximate IID data distribution within the group and accordingly propose a multi-stage training approach to solve the problems caused by data heterogeneity.

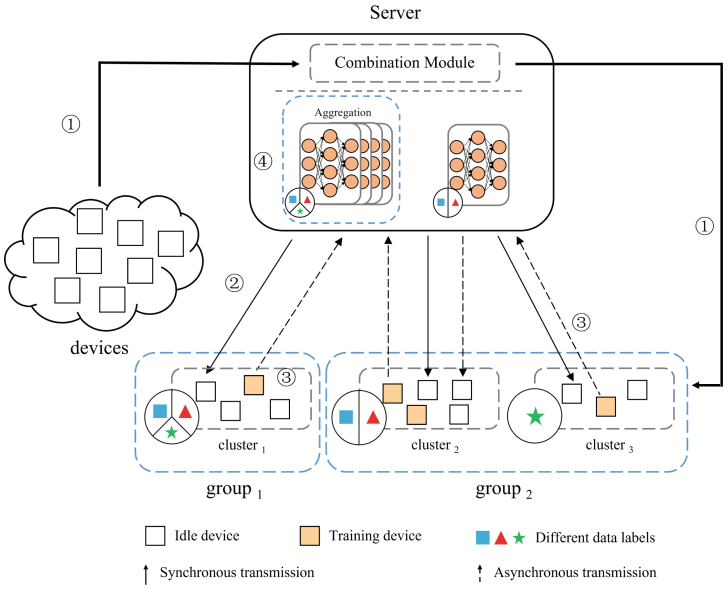


Fig. 1. Overview of MSSA-FL framework.

Combination Module. In order to form a balanced local dataset within a group, the main works of the combination module include: (1) Divide devices with similar data distribution into the same cluster. (2) Divide clusters with complementary data distributions into a group to form a locally balanced dataset.

Device Clustering. Several *cluster federated learning* (CFL) schemes [26–28] have been proposed infer whether customers have similar data distributions from updates of their local models. In this section, we follow the client clustering approach in [26], using the cosine similarity of the clients' weight updates as the proximities of the local optimizations. The cosine similarity between the updates of any two devices i and j is defined as follows:

$$CS(i,j) = \frac{\langle \Delta\omega^{(i)}, \Delta\omega^{(j)} \rangle}{||\Delta\omega^{(i)}||\ ||\Delta\omega^{(j)}||} \tag{1}$$

In our approach, parameter updates are obtained for the local model of each device through one round pre-training. To reduce the bias of cosine similarity, the mean of absolute differences of pairwise cosine similarity is used as the dissimilarity measure for clustering, as follows:

$$MADC(i,j) = \frac{1}{n-2} \sum_{z \neq i,j} |CS(i,z) - CS(j,z)| \tag{2}$$

With the above clustering method, the server divides all devices into multiple clusters, and the devices in each cluster have similar data distribution. As

shown in Fig. 1, there are three clusters and each cluster has multiple devices with similar data distribution.

Cluster Grouping. Although devices are clustered, the data distribution of each cluster is still unknown. So we need to accomplish the following two works, inferring the data distribution of each clusters and determining the group member clusters that may achieve data balance within a group.

We leverage the method proposed in [29], using auxiliary dataset to infer the data composition of each cluster. First, after client clustering, server selects a representative device in each cluster to represents the data distribution of its cluster. Next, establish the (3) to infer the composition of training data by finding the relationship between the data of a specific label and the weight change of the output layer.

$$\Delta\omega^{p}_{(p,i)} \cdot \widehat{N}_{p,i} + (\sum_{p=1}^{Q} N^{j}_{p} - \widehat{N}_{p,i}) \frac{\sum_{j=1}^{Q}(\Delta\omega^{j}_{(p,i)}) - \Delta\omega^{p}_{(p,i)}}{Q-1} = n^{p}_{a} \cdot (\omega^{k}_{p,i} - \omega^{G}_{p,i}) \quad (3)$$

where ω^k is the local model of device k after pre-training, ω^G is the initial global model. Q is the number of labels. $\Delta\omega^{j}_{(p,i)}$ is the *i-th* weight update of the *p-th* output node from g_{L_j} and g_{L_j} is the weights update obtained by training the ω^G on the samples of class j in the auxiliary dataset. n^p_a is the sample number of class p in the auxiliary data, $\widehat{N}_p$ is the predicted sample quantity of class p, $\omega^k_{p,i}$ and $\omega^G_{p,i}$ are link weights $\omega_{(p,i)}$ of ω^k and ω^G, respectively. $\sum_{p=1}^{Q} N^j_p$ is the overall number of all samples owned by device k. And we can obtain the final result as the average value of all calculated $\widehat{N}_{p,i}$ (denoted as $\widehat{N}_p$). Finally, cluster's data composition can be obtained after all classes are calculated.

KL divergence is used to measure the similarity of two distributions. The server computes all combinations of clusters and calculates the KL divergence between the data distribution for each combination and the ideal data distribution. Here, the ideal data distribution can be a uniform distribution with balanced labels. The server picks the combination result with the smallest KL divergence until all clusters are included in a certain group. The detail of combination module is described in Algorithm 1.

As illustrated in Fig. 1, since the data distribution of $cluster_1$ is closest to the ideal data distribution, $cluster_1$ forms $group_1$ by itself. Similarly, $cluster_2$ and $cluster_3$ form $group_2$ because their combined data distribution is almost balanced. By combination module, a locally balanced dataset is approximately generated in each group.

Multi-stage Training. A natural motivation to address data heterogeneity in FL is to supplement the imbalanced data in the device so that the local models are trained on IID dataset. Considering the protection of private data, the server transfer the model between multiple devices and perform multiple trainings to solve the local data Non-IID problem in FL.

It is significant to identify the devices that can form a balanced data set. Thus, we present the combination module described above, whose result guide

Algorithm 1: Device Grouping Algorithm

Input: the number of clusters $n_{clusters}$, uniform distribution P_u, global initial model ω_{init}. D_{kl} is Kullback-Leibler divergence.
Output: the grouping result *Group*
Get all devices' weight updates by a round pre-training.
$CS \leftarrow$ Cosine similarity between any two devices according to (1)
$MADC \leftarrow$ Calculate MADC(CS) using (2)
$S_{cluster} \leftarrow$ Hierarchical Clustering($MADC$, $n_{clusters}$)
$S_{dis} \leftarrow$ Estimates the empirical data distribution of clusters according to (3)
$S_{com} \leftarrow$ COMBINATIONS($S_{cluster}, S_{dis}$)
while $\mid S_{cluster} \mid > 0$ **do**
  $g = \arg\min_k(D_{kl}(P_k||P_u))$, $k \in S_{com}$
  for *cluster in g* **do**
    $S_{cluster} \leftarrow S_{cluster} - cluster$
  end
  if $|S_{cluster}|$ *is changed* **then**
    $Group \leftarrow Group + g$
  end
end
return *Group*

the determination of which devices compose the locally IID dataset. In the combination module, devices with similar data distribution generate a cluster, and the data distribution of devices within the cluster can be further inferred. Each cluster appears to be a virtual local device whose data distribution is known. Accordingly, the combination module is capable of grouping clusters with complementary data distributions so that a locally balanced dataset is generated within each group. After completing all stages of training in a group, the local model is equivalent to being trained on the IID dataset, which narrows the gap between local models and reduces the bias introduced to the global model during aggregation in server.

Simultaneously, based on combination module, multiple devices in the same cluster are assigned the same function to provide that stage of training. A device can release training resources after completing local training without waiting for the current model to complete all stages, allowing such devices to immediately provide training for other intra-group models. As a result, the training resources for intra-group models are more flexibly satisfied and the device utilization is improved because of the opportunity for training multiple models.

3.3 Semi-asynchronous Training

Although the multi-stage training strategy can solve the problem of data heterogeneity, it poses a challenge for system efficiency. Compared to single training for each client per round in FedAvg, multi-stage training makes models train multiple times before aggregation, resulting in extended training time per round.

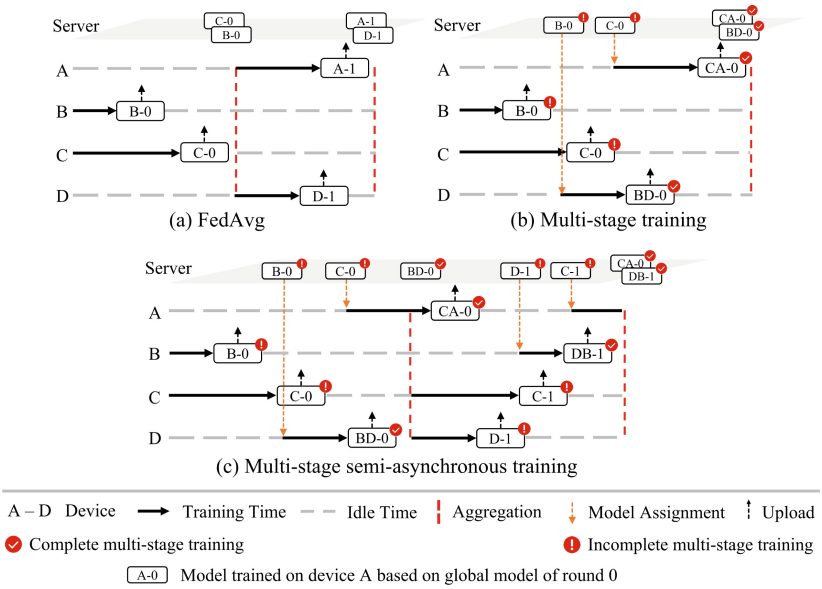


Fig. 2. Illustration of different FL operating principles. Suppose device A and device B are in one cluster, device C and device D are in another cluster. These two clusters form a group.

Furthermore, device heterogeneity may lead longer device idle time when synchronizing updates.

Considering the degradation on efficiency of multi-stage training, we adopt a compromise solution, a semi-asynchronous update approach. In our semi-asynchronous update approach, the server will aggregate the models in the following two cases: (1) A specified number of models complete multi-stage training in the current round. (2) The time limit has been reached in the current round. Consequently, the server only synchronously aggregates models that have completed multi-stage training within the round time. Residual models will keep training asynchronously to participate in aggregation on subsequent rounds.

Figure 2 depicts above synchronous and semi-asynchronous algorithms' operating principles. In FedAvg, only a few devices are selected to train in each round, and the aggregation of models occurs as the slowest device completes its local training. As for multi-stage training, although all devices participate in the training, the round time increases significantly due to the multiple trainings per model (e.g., $B \rightarrow D$, $C \rightarrow A$). With that comes longer wait time for idle devices. In the semi-asynchronous training, because of exceeding the round time limit, the server updates the global model in advance and adds new models in the next round. Thus, semi-asynchronous update increases the frequency of global model updates and makes more models to be trained in a round, providing more devices the opportunity to overtrain.

3.4 Model Assignment

In this section, we propose an assignment strategy for intermediate model which have not completed all stages of training to select the suitable device for next training stage.

Training Sequence Optimization. We observed that the model loses part of the knowledge learned previously when it is trained on a new dataset, so it can be inferred that the last stage of the multi-stage training has the largest contribution to the model. Inspired by the above, we propose a method to determine the order of multi-stage training based on the recognition ability of the global model for different labels.

First, we use recall rate, which can be obtained through the validation of the global model on server, to measure the ability of the model to recognize a certain label samples. Then, we adopt the idea of z-score based outlier detection and consider classes with scores grater than the threshold to be low-accuracy labels and add them to set S_{low}. For a intermediate model, we assume $S_{ls} = [c_1, c_2, ..., c_g]$ as the clusters corresponding to its lacking stages. We add the data proportions of classes in S_{low} to get $[q_1, q_2, \ldots, q_g]$:

$$q_j = \sum_{l \in S_{low}} Dis_j^l, j \in S_{ls} \tag{4}$$

where Dis_j^l is the proportion of data with label l of cluster j. Clusters with larger q-value usually have more data samples with low-accuracy labels and they are should be trained in the back of the queue, for enhancing the absorption of more difficult knowledge. To maintain stochasticity in the order, we randomly select the next stage of clusters among the $\lceil g/2 \rceil$ clusters with the smallest q.

Device Selection. Device heterogeneity causes a training speed difference among devices. By increasing the probability of being selected for devices with short training time, we expect to involve more high-speed devices in training to get more models aggregated in less time. The probability of a idle device k to be selected is:

$$p_k = \frac{\sum_{j \in IC} \frac{1}{t_j}}{t_k} \tag{5}$$

where IC is the set of idle devices that may be selected for the next stage of the model and t_k is the training time of device k which can be taken from historical performance. To prevent fast devices from being selected frequently and introducing bias, we set a *credit* for all device in each federated round. The *credit* decreases as the device is selected, and the server will not select devices without any *credit*.

3.5 Model Aggregation

Considering the problem that local models accomplished in the same round may originate from different versions of the global model in the asynchronous update

method. As shown in Fig. 2(c), the server collects two models with different versions (*CA-0* and *DB-1*) at the end of second round. In this section, we modify the original aggregation method to reduce the influence of staleness models on the global model.

The server maintains a global model version τ_g, which is incremented when completing aggregation. And the version of the model k, τ_k, is defined as the current τ_g when it is first sent to a device. Consequently, the staleness of an aggregated model k is:

$$a_k = \tau_g - \tau_k. \tag{6}$$

Our motivation is to assign larger aggregate weights to models with smaller staleness to support fresher local updates. Thus, we define the aggregated weights of model k as:

$$p_k(t) = \frac{|N_k|\gamma^{a_k(t)}}{\sum_{i \in M(t)} |N_i|\gamma^{a_i(t)}} 1(k \in M(t)) \tag{7}$$

where $|N_k|$ is the data size of model k during training, $a_k(t)$ is the staleness of model k at round t and $M(t)$ is the set of models to be aggregated at round t. Here, γ is a real-valued constant factor less than 1.

Eventually, the global model of MSSA-FL will be updated according to the following equation:

$$\omega^{t+1} = \sum_{k \in M(t)} p_k(t) \times \omega_k^t \tag{8}$$

4 Experiment

4.1 Experimental Setup

We implement our proposed MSSA-FL and verify the effectiveness of our proposed method by comparison and analysis.

Datasets and Models. We adopt two widely used public data sets, MNIST and CIFAR-10, and use convolutional neural networks (CNN) for image classification task. The CNN network for MNIST contains: two 5×5 convolutional layers (the first has 10 channels, the second has 20, each followed with 2×2 max pooling), a fully connected layer with 50 units and ReLu activation, and a final output layer. As for CIFAR-10, the model consists of three 3×3 convolution layer with 128 channels and a fully connected layer with 10 units.

Implementation Details. As for the data heterogeneity between devices, we follow the similar method in [30] and use a Dirichlet distribution $Dir(\alpha)$ to generate data partitions for devices, where α is a hyperparameter that can determine the degree of Non-IID data partition. In order to simulate the heterogeneity of devices, we assume that the performance of the devices follow the Gaussian distribution $\mathcal{N}$(a, b)($a = 0.5$ and $b = 0.2$). Meanwhile, we define the performance

of a device as the number of batches it can process per second in training. We ignore the communication latency between devices and the server.

Baseline Algorithms. In order to exhibit the superiority of MSSA-FL, we reimplement other FL schemes for comparison: FedAvg [18], FedProx [8] and FedAsync [23], which are representative synchronous and asynchronous FL protocols. We set μ to 1 for FedProx and keep the maximum staleness of the model at 4 for FedAsync. The settings of other hyperparameters are shown in Table 1.

Table 1. Experimental setup

Parameter	Symbol	MNIST	CIFAR10
Total number of devices	N	100	100
Device selection level	C	0.1	0.1
Local epochs	E	5	5
Mini-batch size	B	50	32
Learning rate	η	0.001	0.005
Maximum time per round (s)	t	400	500
Running time (s)	T	30,000	150,000

4.2 Experimental Results

Accuracy Evaluation. Table 2 and Fig. 3 illustrate the experimental results on convergence accuracy with representative α values. impr. (a) and impr. (b) are the accuracy improvement of MSSA-FL compared with the best and worst baseline FL Algorithm, respectively.

Table 2. Comparison of the best prediction accuracy to baseline approaches.

Dataset	MNIST			CIFAR-10		
	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$	$\alpha = 0.1$	$\alpha = 1$	$\alpha = 10$
FedAvg	76.55	91.89	92.28	53.09	68.33	70.42
FedProx	81.02	91.74	92.07	51.36	67.69	70.17
FedAsync	88.13	92.34	92.86	55.19	68.08	68.52
MSSA-FL	90.42	93.71	93.92	59.35	70.93	71.02
Impr. (a)	2.29%	1.37%	1.06%	4.16%	2.60%	0.60%
Impr. (b)	13.87%	1.97%	1.85%	7.99%	3.24%	2.50%

Under all experimental settings, MSSA-FL has the highest model accuracy. It can be seen a greater performance advantage of MSSA-FL as the α decreases. In

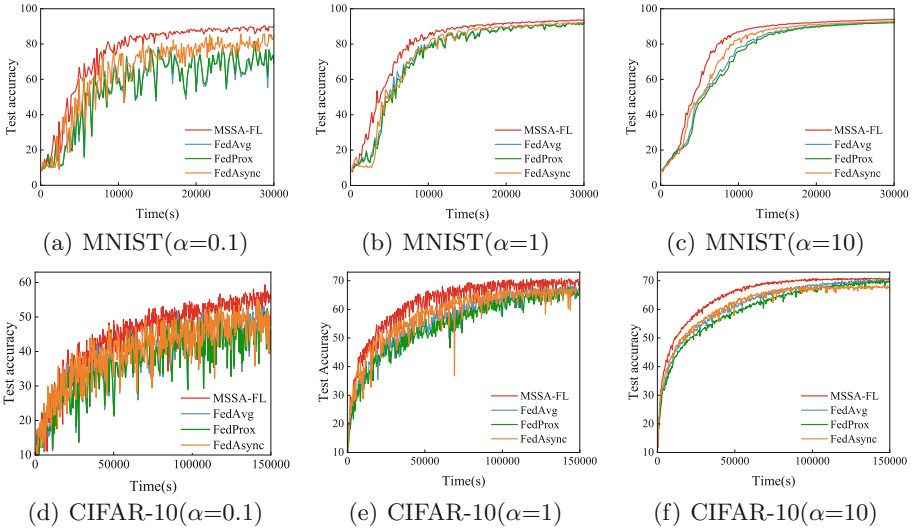


Fig. 3. Performance comparison of different FL algorithms on MNIST and CIFAR-10 with $\alpha = 0.1$, $\alpha = 1$ and $\alpha = 10$.

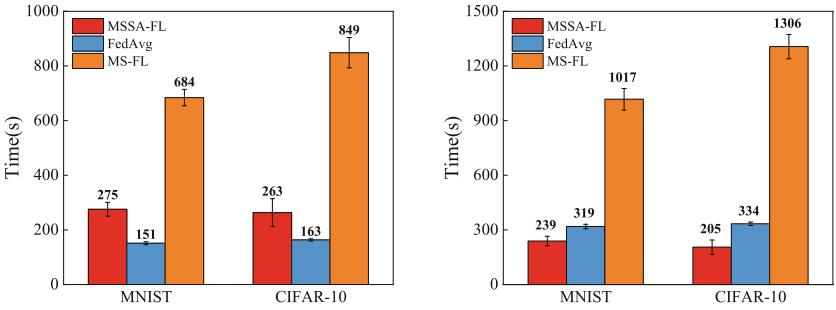
the MNIST with $\alpha = 0.1$, MSSA-FL outperforms the best baseline FL algorithm, FedAsync, by 2.29%, and the worst baseline algorithm, FedAvg, by 13.87%. In the MNIST with $\alpha = 10$, MSSA-FL outperforms the best baseline FL algorithm, FedAsync, by 1.06%, and the worst baseline algorithm, FedProx, by 1.85%. There is the similar performance result in CIFAR-10. MSSA-FL improves the prediction performance by 6.56%, 2.60%, 0.60% compared to FedAvg with $\alpha = 0.1$, $\alpha = 1$, and $\alpha = 10$. The above results demonstrate that the multi-stage training strategy in MSSA-FL solve the data heterogeneity problem and improves the accuracy of the model. When α is large, MSSA-FL generates more groups with a single cluster (only a single cluster in a group is the best grouping result). Therefore, MSSA-FL would degenerate into a algorithm similar to FedAvg.

Efficiency Evaluation. Table 3 shows the time required to reach the target accuracy when specific α value ($\alpha = 0.1$). In MNIST, MSSA-FL saves 68.56% and 67.98% time over FedAvg and FedProx. As for CIFAR-10, MSSA-FL saves 22.50% and 47.98% time over FedAvg and FedProx. FedAvg and FedProx use the same synchronous update method, so they always take longer to reach the target accuracy. FedAsync outperforms the above synchronous methods in terms of time efficiency. By contrast, MSSA-FL still reduces the training time by 17.06% and 12.37% in MNIST and CIFAR-10.

Figure 4 shows the average idle time per round of devices in different algorithms under homogeneous and heterogeneous device settings. MS-FL is the multi-stage synchronous federated learning described in Sect. 3.3. It can be seen that MSSA-FL has minimal device idle time in heterogeneous setting and also

Table 3. Time(s) to achieve the target accuracy.

Algorithms	MNIST (Acc.= 0.8)	CIFAR-10 (Acc.= 0.5)
FedAvg	27536.14	90590.97
FedProx	27036.13	134954.31
FedAsync	10439.65	80113.95
MSSA-FL	8658.13	70205.36

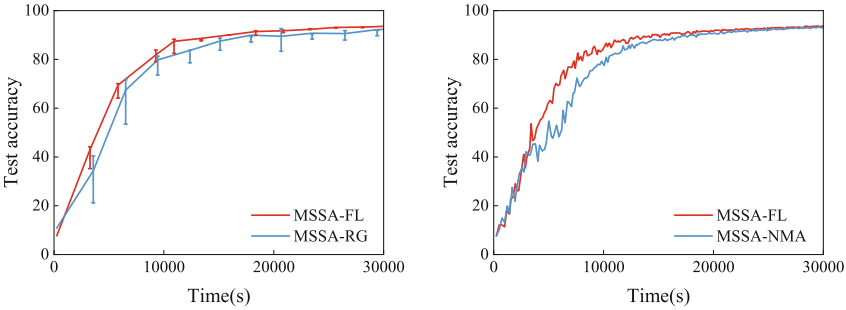


(a) Homogeneous devices. (b) Heterogeneous devices.

Fig. 4. Average device idle time per round.

good performance in homogeneous setting. MSSA-FL allows faster devices train more frequently, which lessens the time of multi-stage training and lets more devices to train in a round. Though FedAvg achieves the shortest device idle time in homogeneous setting, when devices are heterogeneous, slow devices severely block FedAvg training. Due to the longest round time, MS-FL always has the largest device idle time.

Effectiveness of Combination Module and Model Assignment Strategy. We implement two variants of MSSA-FL, MSSA-RG (MSSA-FL with random groups) and MSSA-NMA (MSSA-FL without model assignment policy). Figure 5 shows the comparison results of MSSA-FL and MSSA-RG, MSSA-NMA in MINST with $\alpha = 1$. In Fig. 5(a), we sample the results every 15 rounds and calculate the accuracy of the model formed in each group at the sampling points (the range of accuracy of different groups' model is marked by error lines). MSSA-FL achieves a faster convergence speed and higher accuracy than MSSA-RG. At the same time, the variance of model accuracy for different groups in MSSA-FL is smaller than MSSA-RG, demonstrating the effectiveness of the combination module. As shown in Fig. 5(b), MSSA-NMA achieves the accuracy as high as MSSA-FL. However, it takes 8835.3 s to reach 80% accuracy, which is 13.46% longer than MSSA-FL (7646.41 s). Consequently, model assignment strategy is able to speed up the convergence of the model.



(a) MSSA-FL v.s. MSSA-RG (b) MSSA-FL v.s. MSSA-NMA

Fig. 5. Efficiency evaluation of proposed methods.

5 Conclusion

In this paper, we have proposed a novel FL framework, MSSA-FL to address the heterogeneous data issues and improve the training efficiency. By combination module, MSSA-FL enables to infer devices data distribution and group devices for the purpose of following multi-stage training. MSSA-FL requires the models trained in multi-stage within the group to achieve approximately IID dataset training effect, reducing the impact of Non-IID data among devices. For boosting training efficiency, MSSA-FL adopts the semi-asynchronous update method. Furthermore, we designed a model scheduling strategy and a heuristic aggregation method to further improve the performance of MSSA-FL. We conducted extensive experimental validation and demonstrated that MSSA-FL has the highest prediction performance and fastest convergence speed compared with advanced synchronous and asynchronous FL algorithms.

References

1. Li, E., Zhou, Z., Chen, X.: Edge intelligence: on-demand deep learning model co-inference with device-edge synergy. In: Workshop on Mobile Edge Communication, pp. 31–36 (2018)
2. Zhou, Z., Chen, X., Li, E., et al.: Edge intelligence: paving the last mile of artificial intelligence with edge computing. Proc. IEEE **107**(8), 1738–1762 (2019)
3. Qiu, H., Zheng, Q., et al.: Topological graph convolutional network-based urban traffic flow and density prediction. IEEE Trans. on Intel. Transpor. Syst. **22**(7), 4560–4569 (2020)
4. Li, Y., Song, Y., et al.: Intelligent fault diagnosis by fusing domain adversarial training and maximum mean discrepancy via ensemble learning. IEEE TII. **17**(4), 2833–2841 (2020)
5. Hu, F., Lakdawala, S., Hao, Q., Qiu, M.: Low-power, intelligent sensor hardware interface for medical data preprocessing. IEEE Trans. Info. Tech. Biomed. **13**(4), 656–663 (2009)

6. Li, X., Huang, K., Yang, W., Wang, S., Zhang, Z.: On the convergence of fedavg on non-iid data (2019). arXiv preprint, arXiv:1907.02189
7. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-iid data (2018). arXiv preprint, arXiv:1806.00582
8. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. Proc. Mach. Learn. Sys. **2**, 429–450 (2020)
9. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T.: Scaffold: Stochastic controlled averaging for federated learning. In: International Conference on Machine Learning, pp. 5132–5143 (2020)
10. Duan, M., Liu, D., et al.: Self-balancing federated learning with global imbalanced data in mobile systems. IEEE TPDS **32**(1), 59–71 (2020)
11. Qiu, M., Xue, C., Shao, Z., Sha, E.: Energy minimization with soft real-time and DVS for uniprocessor and multiprocessor embedded systems. In: IEEE DATE, pp. 1–6 (2007)
12. Qiu, M., Liu, J., et al.: A novel energy-aware fault tolerance mechanism for wireless sensor networks. In: IEEE/ACM Conference on GCC (2011)
13. Qiu, M., et al.: Heterogeneous real-time embedded software optimization considering hardware platform. In: ACM Symposium on Applied Computing, pp. 1637–1641 (2009)
14. Lu, Z., et al.: IoTDeM: an IoT Big Data-oriented MapReduce performance prediction extended model in multiple edge clouds. JPDC **118**, 316–327 (2018)
15. Liu, M., Zhang, S., et al.: H infinite state estimation for discrete-time chaotic systems based on a unified model. In: IEEE SMC (B) (2012)
16. Qiu, H., Zheng, Q., et al.: Deep residual learning-based enhanced jpeg compression in the internet of things. IEEE TII **17**(3), 2124–2133 (2020)
17. Qiu, H., Qiu, M., Lu, Z.: Selective encryption on ECG data in body sensor network based on supervised machine learning. Infor. Fusion **55**, 59–67 (2020)
18. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: Artificial Intelligence Statistics, pp. 1273–1282 (2017)
19. Wu, G., Zhang, H., et al.: A decentralized approach for mining event correlations in distributed system monitoring. JPDC **73**(3), 330–340 (2013)
20. Qiu, L., et al.: Optimal big data sharing approach for tele-health in cloud computing. In: IEEE SmartCloud, pp. 184–189 (2016)
21. Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., Khazaeni, Y.: Federated learning with matched averaging (2020). arXiv preprint, arXiv:2002.06440
22. Nishio, T., Yonetani, R.: Client selection for federated learning with heterogeneous resources in mobile edge. In: IEEE ICC, pp. 1–7 (2019)
23. Xie, C., Koyejo, S., Gupta, I.: Asynchronous federated optimization (2019). arXiv preprint, arXiv:1903.03934
24. Wu, W., He, L., Lin, W., et al.: Safa: a semi-asynchronous protocol for fast federated learning with low overhead. IEEE Trans. Comput. **70**(5), 655–668 (2020)
25. Xu, Z., Yu, F., Xiong, J., Chen, X.: Helios: heterogeneity-aware federated learning with dynamically balanced collaboration. In: 58th ACM/IEEE DAC, pp. 997–1002 (2021)
26. Duan, M., et al.: Flexible clustered federated learning for client-level data distribution shift. In: IEEE TPDS (2021)
27. Ghosh, A., Chung, J., Yin, D., Ramchandran, K.: An efficient framework for clustered federated learning. Adv. Neural. Inf. Process. Syst. **33**, 19586–19597 (2020)

28. Sattler, F., Müller, K.R., Samek, W.: Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. IEEE Trans. Neural Netw. Learn. Syst. **32**(8), 3710–3722 (2020)
29. Wang, L., Xu, S., Wang, X., Zhu, Q.: Addressing class imbalance in federated learning. In: AAAI Conference, vol. 35, pp. 10165–10173 (2021)
30. Wang, J., Liu, Q., et al.: Tackling the objective inconsistency problem in heterogeneous federated optimization. Adv. Neural Infor. Proc. Syst. **33**, 7611–7623 (2020)