FedASMU: Efficient Asynchronous Federated Learning with Dynamic Staleness-Aware Model Update

Ji Liu[1*] Juncheng Jia[2,3*], Tianshi Che[4], Chao Huo[2], Jiaxiang Ren[4], Yang Zhou[4], Huaiyu Dai[5], Dejing Dou[6]

[1] Hithink RoyalFlush Information Network Co., Ltd., China.
[2] Soochow University, China
[3] Collaborative Innovation Center of Novel Software Technology and Industrialization, China
[4] Auburn University, United States
[5] North Carolina State University, United States
[6] Boston Consulting Group, China
jiliuwork@gmail.com, jiajuncheng@suda.edu.cn

Abstract

As a promising approach to deal with distributed data, Federated Learning (FL) achieves major advancements in recent years. FL enables collaborative model training by exploiting the raw data dispersed in multiple edge devices. However, the data is generally non-independent and identically distributed, i.e., statistical heterogeneity, and the edge devices significantly differ in terms of both computation and communication capacity, i.e., system heterogeneity. The statistical heterogeneity leads to severe accuracy degradation while the system heterogeneity significantly prolongs the training process. In order to address the heterogeneity issue, we propose an Asynchronous Staleness-Aware Model Update FL framework, i.e., FedASMU, with two novel methods. First, we propose an asynchronous FL system model with a dynamical model aggregation method between updated local models and the global model on the server for superior accuracy and high efficiency. Then, we propose an adaptive local model adjustment method by aggregating the fresh global model with local models on devices to further improve the accuracy. Extensive experimentation with 6 models and 5 public datasets demonstrates that FedASMU significantly outperforms baseline approaches in terms of accuracy (0.60% to 23.90% higher) and efficiency (3.54% to 97.98% faster).

Introduction

In recent years, numerous edge devices have been generating large amounts of distributed data.Due to the implementation of laws and regulations, e.g., General Data Protection Regulation (GDPR) (EU 2018), the traditional training approach, which aggregates the distributed data into a central server or a data center, becomes almost impossible. As a promising approach, Federated Learning (FL) (Kairouz, McMahan, and et al. 2021; Liu et al. 2022a) enables collaborative model training by transferring gradients or models instead of raw data. FL avoids privacy or security issues incurred by direct raw data transfer while exploiting multiple edge devices to train a global model. FL has been applied in diverse areas, such as computer vision (Liu et al. 2020), nature language processing (Liu et al. 2021), bioinformatics (Chen et al. 2021), and healthcare (Nguyen et al. 2022a).

Traditional FL commonly uses a parameter server (server) (Liu et al. 2023a) to manage training on various devices synchronously (McMahan et al. 2017; Li et al. 2020; Liu et al. 2023b; Jia et al. 2023). This synchronous process involves multiple rounds, each with five steps: 1) device selection by the server (Shi et al. 2020), 2) broadcasting the global model to chosen devices, 3) local training on each device, 4) uploading updated models (gradients) to the server, and 5) the server aggregating these models to update the global model after all devices complete the first four steps. Although straightforward, this mechanism can be inefficient due to stragglers, especially with heterogeneous devices (Jiang et al. 2022; Lai et al. 2021). Faster devices often idle while waiting for slower devices, leading to efficiency loss (Wu et al. 2020).

In federated learning (FL), device heterogeneity is notable in both computational and communication capacities (Wu et al. 2020; Che et al. 2022, 2023b), and in data distribution (McMahan et al. 2017; Li et al. 2020; Wang et al. 2020; Che et al. 2023a). This results in varying local training and model update times. Some devices, due to limited bandwidth or high latency, struggle to upload their models promptly, a challenge termed as system heterogeneity. Additionally, the data on each device often exhibits non-Independent and Identically Distributed (non-IID) characteristics, leading to statistical heterogeneity. This can cause divergent local objectives (Wang et al. 2020) and client drift (Karimireddy et al. 2020; Hsu, Qi, and Brown 2019), thereby impacting the accuracy of the global FL model.

Asynchronous FL (Xu et al. 2021; Wu et al. 2020; Nguyen et al. 2022a) allows server-side model aggregation without waiting for slower devices. However, it faces potential accuracy issues due to stale model uploads and non-IID data (Zhou et al. 2021a). For instance, a device might upload a model based on an outdated global model, causing the current global model to regress, leading to lower accuracy. Additionally, without proper staleness control (Xie, Koyejo, and Gupta 2019), asynchronous FL might struggle to converge (Su and Li 2022).

*Corresponding authors.

Current studies tackle system and statistical heterogeneity in isolation. Device scheduling methods (Shi et al. 2020; Shi, Zhou, and Niu 2020; Wu et al. 2020; Zhou et al. 2022; Liu et al. 2022b) address system heterogeneity but may lead to lower accuracy due to limited device participation. Asynchronous FL, addressing system heterogeneity, either uses static polynomial formulas for staleness (Xie, Koyejo, and Gupta 2019; Su and Li 2022; Chen, Mao, and Ma 2021) or basic attention mechanisms (Chen et al. 2020), lacking dynamic adjustment in model aggregation, affecting accuracy. Conversely, methods like regularization (Li et al. 2020), gradient normalization (Wang et al. 2020), and momentum techniques (Hsu, Qi, and Brown 2019; Jin et al. 2022a) focus on statistical heterogeneity within synchronous FL.

In this paper, we propose an original Asynchronous Federated learning framework with Staleness-Aware Model Update (FedASMU). To address the system heterogeneity, we design an asynchronous FL system and propose a dynamical adjustment method to update the importance of updated local models and the global model based on both the staleness and the local loss for superior accuracy and high efficiency. We enable devices to adaptively aggregate fresh global models so as to reduce the staleness of the local model. We summarize the major contributions in this paper as follows:

- We propose a novel asynchronous FL system model with a dynamic model aggregation method on the server, which adjusts the importance of updated local models and the global model based on the staleness and the impact of local loss for superior accuracy and high efficiency.
- We propose an adaptive local model adjustment method on devices to integrate fresh global models into the local model so as to reduce staleness for superb accuracy. The model adjustment consists of a Reinforcement Learning (RL) method to select a proper time slot to retrieve global models and a dynamic method to adjust the local model aggregation.
- We conduct extensive experiments with 9 state-of-the-art baseline approaches, 6 typical models, and 5 public real-life datasets, which reveals FedASMU can well address the heterogeneity issues and significantly outperforms the baseline approaches.

The rest of the paper is organized as follows. We present the related work in Section 2. Then, we formulate the problem and explain the system model in Section 3. We propose the staleness-aware model update in Section 4. The experimental results are given in Section 5. Finally, Section 6 concludes the paper.

Related Work

Parallel, distributed, and federated learning have been extensively studied in recent years (Chen, Zhou, and Zhou 2023; Chen et al. 2023; Lee et al. 2019; Wu et al. 2021; Goswami et al. 2020; Zhang et al. 2021; Guo et al. 2022; Yan et al. 2022a; Yan, Zhou, and Guo 2022; Yan et al. 2022b; Jin et al. 2022b, 2021; Zhao et al. 2021; Zhou and Liu 2013; Lee et al. 2013; Zhang et al. 2013; Zhou et al. 2014; Zhang et al. 2014; Bao et al. 2015; Zhou et al. 2015a,b; Lee et al. 2015; Jiang et al. 2019; Zhou 2017; Hong et al. 2023; Chen et al. 2018b,a). A bunch of FL approaches (McMahan et al. 2017; Li et al. 2020; Wang et al. 2020; Karimireddy et al. 2020; Acar et al. 2021) focus on collaboratively training a global model using data from mobile devices, typically employing a synchronous mechanism for server-side model aggregation. They are inefficient due to the straggler effect, where the server waits for all selected devices to upload their models. The issue is exacerbated as device scale and system heterogeneity increase.

Three types of strategies are employed in synchronous FL to manage system heterogeneity: device scheduling (Shi et al. 2020; Shi, Zhou, and Niu 2020; Wu et al. 2020), which may affect accuracy by limiting less capable devices; pruning (Zhang et al. 2022) and dropout (Horvath et al. 2021), potentially leading to reduced accuracy; and device clustering (Li et al. 2022) using hierarchical architectures (Abad et al. 2020), which can compromise efficiency and accuracy due to statistical heterogeneity.

Multiple model aggregation methods (Karimireddy et al. 2020) exists to handle the statistical heterogeneity with the synchronous mechanism. In particular, regularization (Li et al. 2020; Acar et al. 2021), gradient normalization (Wang et al. 2020), classifier calibration (Luo et al. 2021), and momentum-based (Hsu, Qi, and Brown 2019) methods adjust the local objectives to reduce the accuracy degradation brought by heterogeneous data. Contrastive learning (Li, He, and Song 2021), personalization (Sun et al. 2021), meta-learning-based method (Khodak, Balcan, and Talwalkar 2019), and multi-task learning (Smith et al. 2017) adapt the global model or local models to non-IID data. However, these methods lack dynamic adjustment for model diversity and are limited to synchronous FL.

To conquer the system heterogeneity problem, asynchronous FL (Xu et al. 2021; Nguyen et al. 2022a) enables the global model aggregation without waiting for all the devices. The asynchronous FL can be performed once a model is uploaded from an arbitrary device (Xie, Koyejo, and Gupta 2019) or when multiple models are buffered (Nguyen et al. 2022b). However, the old uploaded models may drag the global model to a previous status, which significantly degrades the accuracy (Su and Li 2022). Several methods are proposed to improve the accuracy of asynchronous FL. For instance, the impact of the staleness and the divergence of model updates is considered to adjust the importance of uploaded models (Su and Li 2022), which cannot dynamically adapt the weights based on the training status, e.g., loss values. The attention mechanism and the average local training time are exploited to adjust the weights of uploaded models (Chen et al. 2020) without the consideration of staleness. Strategies like replacing stale models with the latest global model (Wu et al. 2020) can lead to loss of device-specific data, and while staleness-based formulas (Park et al. 2021; Xie, Koyejo, and Gupta 2019; Chen, Mao, and Ma 2021) and loss value adjustments (Park et al. 2021) are used, they lack dynamic optimization for loss minimization.

Different from the existing works, we propose an asynchronous FL framework, i.e., FedASMU, to address the sys-

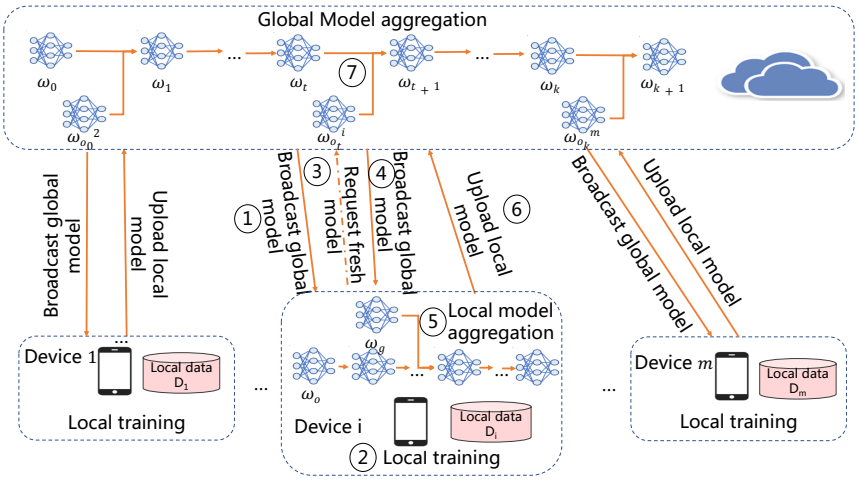


Figure 1: The system model of FedASMU.

tem heterogeneity. FedASMU adjusts the importance of uploaded models based on the staleness while enabling devices to adaptively aggregate fresh global models to further mitigate the staleness issues, which handles the statistical heterogeneity.

Asynchronous System Architecture

In this section, we present the problem formulation for FL and the asynchronous system model.

We consider an FL setting composed of a powerful server and m devices, denoted by $\mathcal{M}$, which collaboratively train a global model. Each device i stores a local dataset $\mathcal{D}_i = \{\boldsymbol{x}_{i,d} \in \mathbb{R}^s, y_{i,d} \in \mathbb{R}\}_{d=1}^{D_i}$ with $D_i = |\mathcal{D}_i|$ data samples where $\boldsymbol{x}_{i,d}$ is the d-th s-dimensional input data vector, and $y_{i,d}$ is the label of $\boldsymbol{x}_{i,d}$. The whole dataset is denoted by $\mathcal{D} = \bigcup_{i\in\mathcal{M}} \mathcal{D}_i$ with $D = \sum_{i\in\mathcal{M}} D_i$. Then, the objective of the training process within FL is:

$$\min_{\boldsymbol{w}} \Big\{ \mathcal{F}(\boldsymbol{w}) \triangleq \frac{1}{|D|} \sum_{i\in\mathcal{M}} |\mathcal{D}_i| \mathcal{F}_i(\boldsymbol{w}) \Big\}, \qquad (\mathcal{P})$$

where $\boldsymbol{w}$ represents the global model, $\mathcal{F}_i(\boldsymbol{w})$ is the local loss function defined as $\mathcal{F}_i(\boldsymbol{w}) \triangleq \frac{1}{|D_i|} \sum_{\{\boldsymbol{x}_{i,d}, y_{i,d}\}\in\mathcal{D}_i} F(\boldsymbol{w}, \boldsymbol{x}_{i,d}, y_{i,d})$, and $F(\boldsymbol{w}, \boldsymbol{x}_{i,d}, y_{i,d})$ is the loss function to measure the error of the model parameter $\boldsymbol{w}$ on data sample $\{\boldsymbol{x}_{i,d}, y_{i,d}\}$.

In order to address the problem defined in Formula $\mathcal{P}$, we propose an asynchronous FL framework as shown in Figure 1. The server triggers the local training of m' devices with a constant time period $\mathcal{T}$. The training process is composed of multiple global rounds. At the beginning of the training, the version of the global model is 0. Then, after each global round, the version of the global model increases by 1. Each global round is composed of 7 steps. First, the server triggers $m'(m' \leq m)$ devices and broadcasts the global model $\boldsymbol{w}_o$ to each device at Step ①. The m' devices are randomly selected available devices. Then, each device performs local training with its local dataset at Step ②. During the local training process, Device i requests a fresh global model (Step ③) from the server to reduce the staleness of the local training as the global model may be updated at the same time. Then, the server sends the global model $\boldsymbol{w}_g$ to the device at Step ④, if $\boldsymbol{w}_g$ is newer than $\boldsymbol{w}_o$, i.e., $g > o$. After receiving the fresh global model, the device aggregates the global model and the latest local model to a new model at Step ⑤ and continues the local training with the new model. When the local training is completed, Device i uploads the local model to the server at Step ⑥. Finally, the server aggregates the latest global model $\boldsymbol{w}_t$ with the uploaded model $\boldsymbol{w}_o^i$ at Step ⑦. When aggregating the global model $\boldsymbol{w}_t$ and the uploaded local model $\boldsymbol{w}_o^i$, the staleness of the local model is calculated as $\tau_i = t - o + 1$. When the staleness τ_i is significant, the local model may drag the global model to a previous version corresponding to inferior accuracy due to legacy information. We discard the uploaded local models when the staleness exceeds a predefined threshold τ to meet the staleness bound so as to ensure the convergence.

Staleness-Aware Model Update

In this section, we propose our dynamic staleness-aware model aggregation method on the server (Step ⑦) and the adaptive local model adjustment method on devices (Steps ③ and ⑤).

Dynamic Model Update on the Server

In this subsection, we propose our dynamic staleness-aware model update method on the server. When the server receives an uploaded model $\boldsymbol{w}_o^i$ from Device i with the original version o, it updates the current global model $\boldsymbol{w}_t$ according to the following formula:

$$\boldsymbol{w}_{t+1} = (1 - \alpha_t^i)\boldsymbol{w}_t + \alpha_t^i \boldsymbol{w}_o^i, \qquad (1)$$

where α_t^i represents the importance of the uploaded model from Device i at global round t, which may have a significant impact on the accuracy of the aggregated model (Xie, Koyejo, and Gupta 2019). Then, we decompose the problem

defined in Formula $\mathcal{P}$ to the following bi-level optimization problem (Bard 1998):

$$\min_{\boldsymbol{w},\boldsymbol{A}} \Big\{ \mathcal{F}(\boldsymbol{w},\boldsymbol{A}) \triangleq \frac{1}{|D|} \sum_{i\in\mathcal{M}} |\mathcal{D}_i| \mathcal{F}_i(\boldsymbol{w}(\boldsymbol{A})) \Big\},$$

$$\boldsymbol{w}(\boldsymbol{A}) = (1-\alpha_t^i)\boldsymbol{w}_t + \alpha_t^i \boldsymbol{w}_o^i, \alpha_t^i \in \boldsymbol{A}, \quad (\mathcal{P}1)$$

$$s.t.\ \boldsymbol{w}_o^i = \underset{\boldsymbol{w}_o^i}{\operatorname{argmin}} \mathcal{F}_i(\boldsymbol{w}_o^i)\ \forall\, i \in \mathcal{M}, \quad (\mathcal{P}2)$$

$$\boldsymbol{A} = \underset{\boldsymbol{A}}{\operatorname{argmin}} \mathcal{F}(\boldsymbol{w},\boldsymbol{A}), \quad (\mathcal{P}3)$$

where $\boldsymbol{A} = \{\alpha_t^1, \alpha_t^2, ..., \alpha_t^m\}$ is a set of values corresponding to the importance of the uploaded models from devices. Problem $\mathcal{P}2$ is the minimization of the local loss function, which is detailed in Section . Inspired by (Xie, Koyejo, and Gupta 2019), we propose a dynamic polynomial function to represent α_t^i defined in Formula 2 for Problem $\mathcal{P}3$.

$$\xi_t^i(o) = \frac{\lambda_t^i}{\sqrt{t}(t-o+1)^{\sigma_t^i}} + \iota_t^i,$$
$$\alpha_t^i(o) = \frac{\mu_\alpha \xi_t^i(o)}{1+\mu_\alpha \xi_t^i(o)}, \quad (2)$$

where μ_α refers to a hyper-parameter, $t-o+1$ represents the staleness, t represents the version of the current global model, o corresponds to the version of the global model that Device i received before local training, λ_t^i, σ_t^i, and ι_t^i are control parameters on Device i at the t-th global round. These three parameters are dynamically adjusted according to Formula 3 to reduce the loss of the global model.

$$\lambda_t^i = \lambda_{o-1}^i - \eta_{\lambda^i} \nabla_{\lambda_{o-1}^i} \mathcal{F}(\boldsymbol{w}_o),$$
$$\sigma_t^i = \sigma_{o-1}^i - \eta_{\sigma^i} \nabla_{\sigma_{o-1}^i} \mathcal{F}(\boldsymbol{w}_o), \quad (3)$$
$$\iota_t^i = \iota_{o-1}^i - \eta_{\iota^i} \nabla_{\iota_{o-1}^i} \mathcal{F}(\boldsymbol{w}_o),$$

where η_{λ^i}, η_{σ^i}, and η_{ι^i} represent the corresponding learning rates for dynamic adjustment, $\nabla_{\lambda_{o-1}^i} \mathcal{F}(\boldsymbol{w}_o)$, $\nabla_{\sigma_{o-1}^i} \mathcal{F}(\boldsymbol{w}_o)$, and $\nabla_{\iota_{o-1}^i} \mathcal{F}(\boldsymbol{w}_o)$ correspond to the respective partial derivatives of the loss function.

The model aggregation algorithm of FedASMU on the server is shown in Algorithm 1. A separated thread periodically triggers m' devices when the number of devices performing training is smaller than a predefined value (Lines 1 - 6). When the server receives $\boldsymbol{w}_o^i$ (Line 8), it verifies if the uploaded model is within the staleness bound (Line 9). If not, the server ignores the $\boldsymbol{w}_o^i$ (Line 10). Otherwise, the server updates the control parameters λ_t^i, σ_t^i, ι_t^i according to Formula 3 (Line 12) and calculates α_t^i based on Formula 2 (Line 13). Afterward, the server updates the global model (Line 14).

Algorithm 1: FedASMU on the Server

Require:
T: The maximum number of global rounds
m': The number of devices to be triggered
τ: The predefined staleness limit
$\mathcal{T}$: The constant time period to trigger devices
$\boldsymbol{w}_0$: The initial global model
λ_0, σ_0, ι_0: The initial control parameters
η_{λ^i}, η_{σ^i}, η_{ι^i}: The learning rates for the dynamic adjustment
Ensure:
$\boldsymbol{w}_T$: The global model at Round T
1: **while** The training is not finished (in parallel) **do**
2: **if** Should trigger new devices **then**
3: Trigger and broadcast the global model to m' devices for parallel local training
4: Sleep $\mathcal{T}$
5: **end if**
6: **end while**
7: **for** t in $\{1, 2, ..., T\}$ **do**
8: Receive $\boldsymbol{w}_o^i$
9: **if** $t - o + 1 > \tau$ **then**
10: Discard $\boldsymbol{w}_o^i$ and continue
11: **else**
12: Update λ_t^i, σ_t^i, ι_t^i according to Formula 3
13: Update α_t^i utilizing Formula 2
14: Update $\boldsymbol{w}_t$ exploiting Formula 1
15: **end if**
16: **end for**

Adaptive Model Update on Devices

In this subsection, we present the local training process with an adaptive local model adjustment method on devices to address Problem $\mathcal{P}2$.

When Device i is triggered to perform local training, it receives a global model $\boldsymbol{w}_o$ from the server and takes it as the initial local model $\boldsymbol{w}_{o,0}$. Within the local training process, the Stochastic Gradient Descent (SGD) approach (Robbins and Monro 1951) is exploited to update the local model based on the local dataset $\mathcal{D}_i$ as defined in Formula 4.

$$\boldsymbol{w}_{o,l} = \boldsymbol{w}_{o,l-1} - \eta_i \nabla \mathcal{F}_i(\boldsymbol{w}_{o,l-1}, \zeta_{l-1}), \zeta_{l-1} \sim \mathcal{D}_i, \quad (4)$$

where o is the version of the global model, l represents the number of local epochs, η_i refers to the learning rate on Device i, and $\nabla \mathcal{F}_i(\cdot)$ corresponds to the gradient based on an unbiased sampled mini-batch ζ_{l-1} from $\mathcal{D}_i$.

In order to reduce the gap between the local model and the global model, we propose aggregating the fresh global model with the local model during the local training process of the devices. During the local training, the global model may be intensively updated simultaneously. Thus, the model aggregation with the fresh global model can well reduce the gap between the local model and the global model. However, it is complicated to determine the time slot to send the request and the weights to aggregate the fresh global model. In this section, we first propose a Reinforcement Learning (RL) method to select a proper time slot. Then, we explain the dynamic local model aggregation method.

Intelligent Time Slot Selection We propose an RL-based intelligent time slot selector to choose a proper time slot to request a fresh global model from the server. In order to reduce communication overhead, we assume only one fresh global model is received during the local training. When the

Algorithm 2: FedASMU on Devices

Require:
t: The number of the meta model update
t_i: The number of local model aggregation
$\mathcal{L}^i$: The maximum number of epochs on Device i
$\boldsymbol{w}_o$: The original global model with Version o
$\boldsymbol{w}_g$: The fresh global model with Version g
θ^i_{t-1}: The parameters of the meta model
$\gamma^i_{t_i-1}$, $\upsilon^i_{t_i-1}$: The control parameters
Ensure:
$\boldsymbol{w}^i_{o,\mathcal{L}}$: The trained local model

$l^* \leftarrow$ Generate a time slot using θ^i_{t-1} or $\mathcal{H}^i_{t_i-1}$
$\boldsymbol{w}^i_{o,0} \leftarrow \boldsymbol{w}_o$
for l in $\{1, 2, ..., \mathcal{L}^i\}$ **do**
  if $l = l^*$ **then**
    Send a fresh global model request to server
    Receive $\boldsymbol{w}_g$
  end if
  if $\boldsymbol{w}_g$ is updated **then**
    $\beta^i_{t_i-1} \leftarrow$ Calculation based on Formula 8
    Update $\boldsymbol{w}_{o,l-1}$ with $\beta^i_{t_i-1}$, $\boldsymbol{w}_g$ and Formula 7
    Update $\gamma^i_{t_i}$ and $\upsilon^i_{t_i}$ based on Formula 9
    $\mathcal{R} \leftarrow loss^{b,i}_{o,l} - loss^{a,i}_{o,l}$
    $b_{t_i} \leftarrow (1-\rho)b_{t_i-1} + \rho\mathcal{R}$
    Update θ_t or $\mathcal{H}^i_{t_i}$ with $\mathcal{R}$
  end if
  Update $\boldsymbol{w}_{o,l}$ based on Formula 4
end for

request is sent early, the server performs few updates and the final updated local model may still suffer from severe staleness. However, when the request is sent late, the local update cannot leverage the information from the fresh global model, corresponding to inferior accuracy. Thus, it is beneficial to choose a proper time slot to send the request.

The intelligent time slot selector is composed of a meta model on the server and a local model on each device. The meta model generates an initial time slot decision for each device, and is updated when a device performs the first local training. The local model is initialized with the initial time slot and updated within the device during the following local training to generate personalized proper time slot for the fresh global model request. We exploit a Long Short-Term Memory (LSTM)-based network with a fully connected layer for the meta model and a $\mathcal{Q}$-learning method (Watkins and Dayan 1992) for each local model. Both the meta model and the local model generate the probability for each time slot. We exploit the ϵ-greedy strategy (Xia and Zhao 2015) to perform the selection.

Within the local training process, we define the reward as the difference between the loss value before model aggregation and that after aggregation. For instance, before aggregating the fresh global model with the request sent after l^* local epochs, the loss value of $\mathcal{F}_i(\boldsymbol{w}_{o,l^*}, \zeta_{l^*})$ is $loss^{b,i}_{o,l^*}$ and that after aggregation is $loss^{a,i}_{o,l^*}$. Then, the reward is $\mathcal{R} = loss^{b,i}_{o,l^*} - loss^{a,i}_{o,l^*}$. Inspired by (Zoph and Le 2017), we update the LSTM model with Formula 5 once an initial aggregation is performed.

$$\theta_t = \theta_{t-1} + \eta_{RL} \sum_{l=1}^{\mathcal{L}} \nabla_{\theta_{t-1}} \log P(\textstyle\int_l | \int_{(l-1):1}; \theta_{t-1})(\mathcal{R} - b_t), \tag{5}$$

where θ_t represents the parameters in the meta model after the t-th meta model update, η_{RL} refers to the learning rate for the training process of RL, $\mathcal{L}$ is the maximum number of local epochs, $\int_l$ corresponds to the decision of sending the request (1) or not (0) after the l-th local epoch, and b_t is a base value to reduce the bias of the model. The model is pre-trained with some historical data and dynamically updated during the training process of FedASMU on each device. The $\mathcal{Q}$-learning method manages a mapping $\mathcal{H}^i$ between the decision and the reward on Device i, which is updated with a weighted average of historical values and reward as shown in Formula 6, inspired by (Dietterich 2000).

$$\mathcal{H}^i_{t_i}(l^*_{t_i-1}, a_{t_i-1}) = \mathcal{H}^i_{t_i-1}(l^*_{t_i-1}, a_{t_i-1}) + \phi(\mathcal{R} + \psi \max_a \mathcal{H}^i_{t_i-1}(l^*_{t_1}, a) - \mathcal{H}^i_{t_i-1}(l^*_{t_i-1}, a_{t_i-1})), \tag{6}$$

where a_{t_i-1} represents the action, $l^*_{t_i-1}$ represents the number of local epochs to send the request within the (t_i-1)-th local model aggregation, ϕ and ψ are hyper-parameters. The action is within an action space, i.e., $a_{t_i-1} \in \{add, stay, minus\}$, with add representing adding 1 epoch to $l^*_{t_i-1}$ ($l^*_{t_i} = l^*_{t_i-1} + 1$), $stay$ representing staying with the same epoch ($l^*_{t_i} = l^*_{t_i-1} + 1$), and $minus$ representing removing 1 epoch from $l^*_{t_i-1}$ ($l^*_{t_i} = l^*_{t_i-1} - 1$).

Dynamic Local Model Aggregation When receiving a fresh global model $\boldsymbol{w}_g$, Device i performs local model aggregation with its current local model $\boldsymbol{w}^b_{o,l}$ using Formula 7.

$$\boldsymbol{w}^a_{o,l} = (1 - \beta^i_{t_i-1})\boldsymbol{w}^b_{o,l} + \beta^i_{t_i-1}\boldsymbol{w}_g, \tag{7}$$

where $\beta^i_{t_i-1}$ is the weight of the fresh global model on Device i at the (t_i-1)-th local global model aggregation. Formula 7 differs from Formula 1 as the received fresh global model corresponds to a higher global version. We exploit Formula 8 to calculate β^i_{t-1}.

$$\phi^i_{t_i-1}(g, o) = \frac{\gamma^i_{t_i-1}}{\sqrt{g}}\left(1 - \frac{\upsilon^i_{t_i-1}}{\sqrt{g - o + 1}}\right),$$
$$\beta^i_{t_i-1}(g, o) = \frac{\mu_\beta \phi^i_{t_i-1}(g, o)}{1 + \mu_\beta \phi^i_{t_i-1}(g, o)}, \tag{8}$$

where μ_β is a hyper-parameter, $\gamma^i_{t_i-1}$ and $\upsilon^i_{t_i-1}$ are control parameters to be dynamically adjusted based on Formula 9.

$$\gamma^i_{t_i} = \gamma^i_{t_i-1} - \eta_{\gamma^i} \nabla_{\gamma^i_{t_i-1}} \mathcal{F}_i(\boldsymbol{w}^b_{o,l}, \zeta_{l-1}),$$
$$\upsilon^i_{t_i} = \upsilon^i_{t_i-1} - \eta_{\upsilon^i} \nabla_{\upsilon^i_{t_i-1}} \mathcal{F}_i(\boldsymbol{w}^b_{o,l}, \zeta_{l-1}), \zeta_{l-1} \sim \mathcal{D}_i, \tag{9}$$

where η_{γ^i} and η_{υ^i} are learning rates for $\gamma^i_{t_i}$ and $\upsilon^i_{t_i}$.

The model update algorithm of FedASMU on devices is shown in Algorithm 2. First, an epoch number l^* (time slot)

Method	LeNet				CNN				ResNet			
	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100		CIFAR-100		Tiny-ImageNet	
	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time
FedASMU	**0.486**	**8800**	**0.182**	**20737**	**0.603**	**10109**	**0.277**	**30569**	**0.358**	**16027**	**0.171**	**22415**
FedAvg	0.431	125514	0.168	95306	0.551	117794	0.243	73145	0.299	109680	0.146	155023
FedProx	0.363	126958	0.172	93430	0.371	/	0.243	73145	0.302	109680	0.148	151935
MOON	0.302	437531	0.172	93430	0.47	100302	0.212	252703	0.302	106021	0.149	139444
FedDyn	0.279	/	0.147	70260	0.507	43974	0.193	52874	0.328	73711	0.142	103661
FedAsync	0.478	36565	0.158	102113	0.491	24931	0.23	37160	0.315	21107	0.143	31288
PORT	0.305	366182	0.104	/	0.385	/	0.145	/	0.314	35712	0.134	78155
ASO-Fed	0.408	83712	0.153	110942	0.482	92246	0.208	103090	0.276	198797	0.122	359899
FedBuff	0.365	9829	0.174	25791	0.364	/	0.201	65736	0.315	27672	0.148	43523
FedSA	0.306	21077	0.0835	/	0.508	20415	0.189	94169	0.195	/	0.116	/

Table 1: The accuracy and training time with FedASMU and diverse baseline approaches. "Acc" represents the convergence accuracy of the global model. "Time" refers to the training time to achieve a target accuracy, i.e., 0.30 for LeNet with CIFAR-10, 0.13 for LeNet with CIFAR-100, 0.40 for CNN with CIFAR-10, 0.15 for CNN with CIFAR-100, 0.25 for ResNet with CIFAR-100, and 0.12 for ResNet with Tiny-ImageNet. "/" represents that the method does not achieve the target accuracy.

to send a request for a fresh global model is generated based on θ_{t-1} when $t = 1$ or $\mathcal{H}^i_{t_i-1}$ when $t \neq 1$ (Line 1). In the l^*-th local epoch, the device sends a request to the server (Line 5), and it waits for the fresh global model (Line 6). After receiving the fresh global model (Line 8), we exploit Formula 8 to update $\beta^i_{t_i-1}$ (Line 9), Formula 7 to update $\boldsymbol{w}_{o,l-1}$ (Line 10), Formula 9 to update γ^i_{t-1} and υ^i_{t-1} (Line 11), the reward values (Line 12), b_{t-1} with ρ being a hyper-parameter (Line 13), θ_t when $t = 1$ or $\mathcal{H}^i_{t_i-1}$ when $t \neq 1$ (Line 14). Finally, the local model is updated (Line 16).

Experiments

In this section, we present the experimental comparison of FedASMU with 9 state-of-the-art approaches. We first present the experimentation setup. Then, we demonstrate the experimental results.

Experimental Setup

We consider an FL environment with a server and 100 heterogeneous devices. We consider both the asynchronous baseline approaches, i.e., FedAsync (Xie, Koyejo, and Gupta 2019), PORT (Su and Li 2022), ASO-Fed (Chen et al. 2020), FedBuff (Nguyen et al. 2022b), FedSA (Chen, Mao, and Ma 2021), and synchronous baseline approaches, i.e., FedAvg (McMahan et al. 2017), FedProx (Li et al. 2020), MOON (Li, He, and Song 2021), and FedDyn (Acar et al. 2021). We utilize 5 public datasets, i.e., Fashion-MNIST (FMNSIT) (Xiao, Rasul, and Vollgraf 2017), CIFAR-10 and CIFAR-100 (Krizhevsky, Hinton et al. 2009), IMDb (Zhou et al. 2021b), and Tiny-ImageNet (Le and Yang 2015). The data on each device is non-IID based on a Dirichlet distribution (Li et al. 2021). We leverage 6 models to deal with the data, i.e., LeNet5 (LeNet) (LeCun et al. 1989), a synthetic CNN network (CNN), ResNet20 (ResNet) (He et al. 2016), AlexNet (Krizhevsky, Sutskever, and Hinton 2012), TextCNN (Zhou et al. 2021b), and VGG-11 (VGG) (Simonyan and Zisserman 2015).

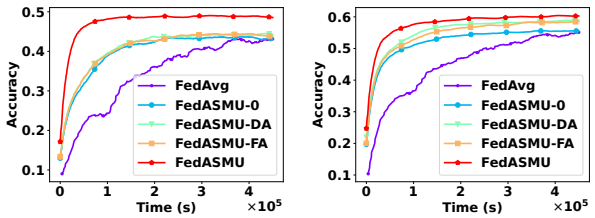
Evaluation of FedASMU

As shown in Tables 1 and 2, FedASMU consistently corresponds to the highest convergence accuracy and training speed. Compared with synchronous baseline approaches, the training speed of FedASMU is much faster than FedAvg (58.23% to 92.01%), FedProx (58.23% to 93.06%), MOON(74.66% to 97.98%), and FedDyn (42.10% to 91.31%) because of asynchronous model update, while the convergence accuracy of FedASMU can still outperform the baseline approaches (0.80% to 16.65% higher for FedAvg, 0.70% to 23.20% higher for FedProx, 0.70% to 18.30% higher for MOON, 0.80% to 18.90% higher for FedDyn). Compared with asynchronous baseline approaches, FedASMU corresponds to the fastest to achieve a target accuracy (6.19% to 84.95% faster than FedAsync, 27.57% to 97.59% faster than PORT, 70.38% to 93.75% faster than ASO-Fed, 10.46% to 69.64% faster than FedBuff, and 3.54% to 67.5% faster than FedSA). In addition, the accuracy of FedASMU is significantly higher (0.70% to 11.70% compared with FedAsync, 0.60% to 21.80% compared with PORT, 2.89% to 13.90% compared with ASO-Fed, and 0.60% to 23.90% compared with FedBuff). The accuracy advantage of FedASMU is brought by the dynamic adjustment of the weights within the model aggregation process on both the server and the devices while the high training speed is because of the asynchronous mechanism and the aggregation of the local model and the fresh global model during the local training process.

We further carry out experimental evaluation with diverse bandwidth, various device heterogeneity, and bigger number of devices. When devices have limited network connection (the bandwidth becomes modest), FedASMU corresponds to slightly higher accuracy (5.04% to 9.34%) and training speed (21.21% to 62.17%) compared with baseline approaches. The advantages of FedASMU become less significant due to extra global model transfer. Although FedASMU introduces more data communication while retrieving fresh

Method	AlexNet				VGG				TextCNN		LeNet	
	CIFAR-10		CIFAR-100		CIFAR-10		CIFAR-100		IMDb		FMNIST	
	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time
FedASMU	**0.490**	**12591**	**0.246**	**12150**	**0.653**	**43093**	**0.264**	**83226**	**0.882**	**3537**	**0.829**	**8250**
FedAvg	0.432	157678	0.205	92558	0.508	335866	0.0975	/	0.874	13960	0.706	65000
FedProx	0.433	141125	0.209	91369	0.505	331991	0.0929	/	0.875	15668	0.708	65000
MOON	0.429	157678	0.202	89297	0.47	335866	0.0991	/	0.875	13960	0.708	65000
FedDyn	0.428	144999	0.197	103950	0.549	190403	0.218	307955	0.874	12674	0.761	40607
FedAsync	0.411	83693	0.203	13717	0.637	45940	0.147	375236	0.875	5837	0.779	12371
PORT	0.365	/	0.192	17400	0.552	75036	0.209	120533	0.876	4884	0.711	75716
ASO-Fed	0.446	55292	0.238	60864	0.533	268349	0.125	405906	0.811	/	0.756	41100
FedBuff	0.469	27763	0.223	27672	0.62	109082	0.238	167053	0.876	7671	0.767	27179
FedSA	0.416	18363	0.176	15933	0.383	/	0.0319	/	0.865	5251	0.783	8553

Table 2: The accuracy and training time with FedASMU and diverse baseline approaches. "Acc" is the convergence accuracy of the global model. "Time" refers to the training time to achieve a target accuracy, i.e., 0.40 for AlexNet with CIFAR-10, 0.12 for AlexNet with CIFAR-100, 0.45 for VGG with CIFAR-10, 0.12 for VGG with CIFAR-100, 0.85 for TextCNN, and 0.70 for LeNet. "/" represents that the method does not achieve the target accuracy.



(a) LeNet (b) CNN

Figure 2: The accuracy and training time with FedASMU, FedASMU-DA, FedASMU-FA, FedASMU-0, and FedAvg on CIFAR-10.

global models, it can well improve the efficiency of the FL training. When the devices are heterogeneous (the diversity of the computation and communication capacity becomes severe), FedASMU performs much better, i.e., the advantages augment 13.67% to 20.10% in terms of accuracy and 85.39% to 91.93% in terms of efficiency. When the devices significantly differ, FedASMU can dynamically adjust the model aggregation on both the server and devices with much better performance. The performance of FedASMU is significantly better than that of the baseline approaches with more devices (4.52% to 15.05% higher in terms of accuracy and 53.47% to 91.20% faster), which demonstrates the excellent scalability of FedASMU.

As shown in Figure 2, we conduct an ablation study with FedASMU-DA, FedASMU-FA, FedASMU-0, and FedAvg. FedASMU-DA represents FedASMU without dynamic model aggregation. FedASMU-FA refers to FedASMU without fresh global model aggregation. FedASMU-0 is FedASMU without the two methods, equivalent to FedAsync with staleness bound. As the dynamic weight adjustment can improve the accuracy, FedASMU outperforms FedASMU-DA (1.38% to 4.32%) and FedASMU-FA outperforms FedASMU-0 (0.65% to 3.04%) in terms of accuracy. As the fresh global model aggregation can reduce the staleness between local models and the global model, FedASMU corresponds to a shorter training time (44.77% to 73.96%) to achieve the target accuracy (0.30 for LeNet and 0.40 for CNN) and higher accuracy (1.75% to 4.71%) compared with FedASMU-FA. In addition, FedASMU-DA leads to better performance (1.04% to 3.41% in terms of accuracy and 15.71% to 19.54% faster) compared with FedASMU-0. Both FedASMU-DA and FedASMU-FA outperform FedAvg in terms of accuracy (0.73% to 3.75%) and efficiency (72.88% to 85.72%). Although FedASMU-0 corresponds to slightly higher accuracy (0.08% to 0.34%) compared with FedAvg, it leads to much higher efficiency (67.84% to 82.26% faster) because of the asynchronous mechanism.

Conclusion

While asynchronous FL can improve the efficiency with heterogeneous devices, staleness may severely degrade the performance. In this paper, we propose a novel Asynchronous Staleness-Aware Model Update FL framework (FedASMU) with an asynchronous system model and two novel methods, i.e., a dynamic model aggregation method on the server and an adaptive local model adjustment method on devices. The adaptive local model adjustment method consists of an RL-based time slot selection method and a dynamic local model aggregation method.Extensive experimentation reveals significant advantages of FedASMU compared with synchronous and asynchronous baseline approaches in terms of accuracy (0.60% to 23.90% higher) and efficiency (3.54% to 97.98% faster).

Acknowledgements

This work is partially sponsored by the National Science Foundation under Grant No. OAC-2313191 (for T. Che and Y. Zhou).

References

Abad, M. S. H.; Ozfatura, E.; Gunduz, D.; and Ercetin, O. 2020. Hierarchical federated learning across heterogeneous cellular networks. In *IEEE ICASSP*, 8866–8870.

Acar, D. A. E.; Zhao, Y.; Matas, R.; Mattina, M.; Whatmough, P.; and Saligrama, V. 2021. Federated Learning Based on Dynamic Regularization. In *ICLR*, 1–36.

Bao, X.; Liu, L.; Xiao, N.; Zhou, Y.; and Zhang, Q. 2015. Policy-driven autonomic configuration management for nosql. In *CLOUD*, 245–252.

Bard, J. F. 1998. *Practical Bilevel Optimization: Algorithms and Applications*. Springer.

Che, T.; Liu, J.; Zhou, Y.; Ren, J.; Zhou, J.; Sheng, V. S.; Dai, H.; and Dou, D. 2023a. Federated Learning of Large Language Models with Parameter-Efficient Prompt Tuning and Adaptive Optimization. In *EMNLP*. Singapore.

Che, T.; Zhang, Z.; Zhou, Y.; Zhao, X.; Liu, J.; Jiang, Z.; Yan, D.; Jin, R.; and Dou, D. 2022. Federated Fingerprint Learning with Heterogeneous Architectures. In *ICDM*, 31–40. IEEE.

Che, T.; Zhou, Y.; Zhang, Z.; Lyu, L.; Liu, J.; Yan, D.; Dou, D.; and Huan, J. 2023b. Fast federated machine unlearning with nonlinear functional theory. In *ICML*, 4241–4268. PMLR.

Chen, M.; Mao, B.; and Ma, T. 2021. FedSA: A staleness-aware asynchronous Federated Learning algorithm with non-IID data. *Future Generation Computer Systems (FGCS)*, 120: 1–12.

Chen, S.; Xue, D.; Chuai, G.; Yang, Q.; and Liu, Q. 2021. FL-QSAR: a federated learning-based QSAR prototype for collaborative drug discovery. *Bioinformatics*, 36(22-23): 5492–5498.

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

Chen, Z.; Feng, G.; Liu, B.; and Zhou, Y. 2018a. Construction policy of network service chain oriented to resource fragmentation optimization in operator network. *JEIT*, 40(4): 763–769.

Chen, Z.; Feng, G.; Liu, B.; and Zhou, Y. 2018b. Delay optimization oriented service function chain migration and re-deployment in operator network. *Acta Electronica Sinica*, 46(9): 2229–2237.

Chen, Z.; Tan, X.; Zhou, Z.; and Zhou, Y. 2023. A channel aggregation based dynamic pruning method in federated learning. In *IEEE Global Communications Conference (GLOBECOM)*. To appear.

Chen, Z.; Zhou, C.; and Zhou, Y. 2023. A hierarchical federated learning model with adaptive model parameter aggregation. *Computer Science and Information Systems*, 20(3): 1037–1060.

Dietterich, T. G. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *JAIR*, 13: 227–303.

EU. 2018. European Union's General Data Protection Regulation (GDPR). https://eugdpr.org/, accessed 2018-1.

Goswami, S.; Pokhrel, A.; Lee, K.; Liu, L.; Zhang, Q.; and Zhou, Y. 2020. GraphMap: scalable iterative graph processing using NoSQL. *The Journal of Supercomputing (TJSC)*, 76(9): 6619–6647.

Guo, G.; Yan, D.; Yuan, L.; Khalil, J.; Long, C.; Jiang, Z.; and Zhou, Y. 2022. Maximal directed quasi-clique mining. In *ICDE*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *IEEE Conf. on CVPR*, 770–778.

Hong, J.; Zhu, Z.; Lyu, L.; Zhou, Y.; Boddeti, V. N.; and Zhou, J. 2023. Int. Workshop on Federated Learning for Distributed Data Mining. In *KDD*, 5861–5862. Long Beach, CA.

Horvath, S.; Laskaridis, S.; Almeida, M.; Leontiadis, I.; Venieris, S.; and Lane, N. 2021. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. *NeurIPS*, 34.

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

Jia, J.; Liu, J.; Zhou, C.; Tian, H.; Dong, M.; and Dou, D. 2023. Efficient Asynchronous Federated Learning with Sparsification and Quantization. *CCPE*. To appear.

Jiang, Y.; Perng, C.-S.; Sailer, A.; Silva-Lepe, I.; Zhou, Y.; and Li, T. 2019. Csm: A cloud service marketplace for complex service acquisition. *ACM TIST*, 8(1): 1–25.

Jiang, Z.; Wang, W.; Li, B.; and Li, B. 2022. Pisces: Efficient Federated Learning via Guided Asynchronous Training. *arXiv*.

Jin, J.; Ren, J.; Zhou, Y.; Lv, L.; Liu, J.; and Dou, D. 2022a. Accelerated Federated Learning with Decoupled Adaptive Optimization. In *ICML*, volume 162, 10298–10322.

Jin, J.; Zhang, Z.; Zhou, Y.; and Wu, L. 2022b. Input-agnostic certified group fairness via gaussian parameter smoothing. In *ICML*.

Jin, R.; Li, D.; Gao, J.; Liu, Z.; Chen, L.; and Zhou, Y. 2021. Towards a better understanding of linear models for recommendation. In *KDD*, 776–785.

Kairouz, P.; McMahan, H. B.; and et al. 2021. Advances and Open Problems in Federated Learning. *Found. Trends Mach. Learn.*

Karimireddy, S. P.; Kale, S.; Mohri, M.; Reddi, S.; Stich, S.; and Suresh, A. T. 2020. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *ICML*, volume 119, 5132–5143.

Khodak, M.; Balcan, M.-F. F.; and Talwalkar, A. S. 2019. Adaptive Gradient-Based Meta-Learning Methods. In *NeurIPS*, 1–12.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *NeurIPS*, 1106–1114.

Lai, F.; Zhu, X.; Madhyastha, H. V.; and Chowdhury, M. 2021. Oort: Efficient federated learning via guided participant selection. In *USENIX Symposium on OSDI*, 19–35.

Le, Y.; and Yang, X. 2015. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7): 3.

LeCun, Y.; Boser, B.; Denker, J.; Henderson, D.; Howard, R.; Hubbard, W.; and Jackel, L. 1989. Handwritten digit recognition with a back-propagation network. In *NeurIPS*, volume 2, 1–9.

Lee, K.; Liu, L.; Ganti, R. L.; Srivatsa, M.; Zhang, Q.; Zhou, Y.; and Wang, Q. 2019. Lightweight indexing and querying services for big spatial data. *IEEE TSC*, 12(3): 343–355.

Lee, K.; Liu, L.; Schwan, K.; Pu, C.; Zhang, Q.; Zhou, Y.; Yigitoglu, E.; and Yuan, P. 2015. Scaling iterative graph computations with graphmap. In *IEEE SC*, 57:1–57:12.

Lee, K.; Liu, L.; Tang, Y.; Zhang, Q.; and Zhou, Y. 2013. Efficient and customizable data partitioning framework for distributed big rdf data processing in the cloud. In *CLOUD*, 327–334.

Li, G.; Hu, Y.; Zhang, M.; Liu, J.; Yin, Q.; Peng, Y.; and Dou, D. 2022. FedHiSyn: A Hierarchical Synchronous Federated Learning Framework for Resource and Data Heterogeneity. In *ICPP*.

Li, Q.; Diao, Y.; Chen, Q.; and He, B. 2021. Federated learning on non-iid data silos: An experimental study. *arXiv:2102.02079*.

Li, Q.; He, B.; and Song, D. 2021. Model-Contrastive Federated Learning. In *CVPR*, 10713–10722.

Li, T.; Sahu, A. K.; Zaheer, M.; Sanjabi, M.; Talwalkar, A.; and Smith, V. 2020. Federated Optimization in Heterogeneous Networks. In *MLSys*, volume 2, 429–450.

Liu, J.; Huang, J.; Zhou, Y.; Li, X.; Ji, S.; Xiong, H.; and Dou, D. 2022a. From distributed machine learning to federated learning: a survey. *KAIS*, 64(4): 885–917.

Liu, J.; Jia, J.; Ma, B.; Zhou, C.; Zhou, J.; Zhou, Y.; Dai, H.; and Dou, D. 2022b. Multi-Job Intelligent Scheduling With Cross-Device Federated Learning. *TPDS*, 34(2): 535–551.

Liu, J.; Wu, Z.; Yu, D.; Ma, Y.; Feng, D.; Zhang, M.; Wu, X.; Yao, X.; and Dou, D. 2023a. Heterps: Distributed deep learning with reinforcement learning based scheduling in heterogeneous environments. *FGCS*, 148: 106–117.

Liu, J.; Zhou, X.; Mo, L.; Ji, S.; Liao, Y.; Li, Z.; Gu, Q.; and Dou, D. 2023b. Distributed and deep vertical federated learning with big data. *CCPE*, e7697.

Liu, M.; Ho, S.; Wang, M.; Gao, L.; Jin, Y.; and Zhang, H. 2021. Federated learning meets natural language processing: A survey. *arXiv preprint arXiv:2107.12603*.

Liu, Y.; Huang, A.; Luo, Y.; Huang, H.; Liu, Y.; Chen, Y.; Feng, L.; Chen, T.; Yu, H.; and Yang, Q. 2020. Fedvision: An online visual object detection platform powered by federated learning. In *AAAI*.

Luo, M.; Chen, F.; Hu, D.; Zhang, Y.; Liang, J.; and Feng, J. 2021. No Fear of Heterogeneity: Classifier Calibration for Federated Learning with Non-IID Data. In *NeurIPS*, 5972–5984.

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

Nguyen, D. C.; Pham, Q.-V.; Pathirana, P. N.; Ding, M.; Seneviratne, A.; Lin, Z.; Dobre, O.; and Hwang, W.-J. 2022a. Federated learning for smart healthcare: A survey. *ACM CSUR*, 55(3): 1–37.

Nguyen, J.; Malik, K.; Zhan, H.; Yousefpour, A.; Rabbat, M.; Malek, M.; and Huba, D. 2022b. Federated Learning with Buffered Asynchronous Aggregation. In *AISTATS*, volume 151, 3581–3607.

Park, J.; Han, D.-J.; Choi, M.; and Moon, J. 2021. Sageflow: Robust federated learning against both stragglers and adversaries. In *NeurIPS*, volume 34, 840–851.

Robbins, H.; and Monro, S. 1951. A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Shi, W.; Zhou, S.; and Niu, Z. 2020. Device scheduling with fast convergence for wireless federated learning. In *IEEE ICC*, 1–6.

Shi, W.; Zhou, S.; Niu, Z.; Jiang, M.; and Geng, L. 2020. Joint device scheduling and resource allocation for latency constrained wireless federated learning. *IEEE TWC*, 20(1): 453–467.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

Smith, V.; Chiang, C.-K.; Sanjabi, M.; and Talwalkar, A. S. 2017. Federated Multi-Task Learning. In *NeurIPS*, volume 30, 1–11.

Su, N.; and Li, B. 2022. How Asynchronous can Federated Learning Be? In *IWQoS*, 1–11.

Sun, B.; Huo, H.; YANG, Y.; and Bai, B. 2021. PartialFed: Cross-Domain Personalized Federated Learning via Partial Initialization. In *NeurIPS*, volume 34, 23309–23320.

Wang, J.; Liu, Q.; Liang, H.; Joshi, G.; and Poor, H. V. 2020. Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. In *NeurIPS*, volume 33, 7611–7623.

Watkins, C. J. C. H.; and Dayan, P. 1992. Technical Note Q-Learning. *Machine Learning*, 8: 279–292.

Wu, S.; Li, Y.; Zhang, D.; Zhou, Y.; and Wu, Z. 2021. Topicka: Generating commonsense knowledge-aware dialogue responses towards the recommended topic fact. In *IJCAI*, 3766–3772.

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

Xia, Z.; and Zhao, D. 2015. Online reinforcement learning by bayesian inference. In *IJCNN*, 1–6.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

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

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

Yan, D.; Qu, W.; Guo, G.; Wang, X.; and Zhou, Y. 2022a. Prefixfpm: A parallel framework for general-purpose mining of frequent and closed patterns. *VLDBJ*, 31(2): 253–286.

Yan, D.; Zhou, Y.; and Guo, G. 2022. Think-like-a-task programming model. *Encyclopedia of Big Data Technologies*.

Yan, D.; Zhou, Y.; Guo, G.; and Liu, H. 2022b. Parallel graph processing. *Encyclopedia of Big Data Technologies*.

Zhang, H.; Liu, J.; Jia, J.; Zhou, Y.; and Dai, H. 2022. FedDUAP: Federated Learning with Dynamic Update and Adaptive Pruning Using Shared Data on the Server. In *IJCAI*, 2776–2782.

Zhang, Q.; Liu, L.; Lee, K.; Zhou, Y.; Singh, A.; Mandagere, N.; Gopisetty, S.; and Alatorre, G. 2014. Improving hadoop service provisioning in a geographically distributed cloud. In *CLOUD*.

Zhang, Q.; Liu, L.; Ren, Y.; Lee, K.; Tang, Y.; Zhao, X.; and Zhou, Y. 2013. Residency aware inter-vm communication in virtualized cloud: Performance measurement and analysis. In *CLOUD*.

Zhang, Z.; Jin, J.; Zhang, Z.; Zhou, Y.; Zhao, X.; Ren, J.; Liu, J.; Wu, L.; Jin, R.; and Dou, D. 2021. Validating the Lottery Ticket Hypothesis with Inertial Manifold Theory. *NeurIPS*, 34.

Zhao, X.; Zhang, Z.; Zhang, Z.; Wu, L.; Jin, J.; Zhou, Y.; Jin, R.; Dou, D.; and Yan, D. 2021. Expressive 1-lipschitz neural networks for robust multiple graph learning against adversarial attacks. In *ICML*, 12719–12735.

Zhou, C.; Liu, J.; Jia, J.; Zhou, J.; Zhou, Y.; Dai, H.; and Dou, D. 2022. Efficient device scheduling with multi-job federated learning. In *AAAI*, 9971–9979.

Zhou, C.; Tian, H.; Zhang, H.; Zhang, J.; Dong, M.; and Jia, J. 2021a. TEA-fed: time-efficient asynchronous federated learning for edge computing. In *ACM Int. Conf. on Computing Frontiers*.

Zhou, Y. 2017. *Innovative Mining, Processing, and Application of Big Graphs*. Ph.D. thesis, Georgia Institute of Technology.

Zhou, Y.; and Liu, L. 2013. Social influence based clustering of heterogeneous information networks. In *KDD*, 338–346.

Zhou, Y.; Liu, L.; Lee, K.; Pu, C.; and Zhang, Q. 2015a. Fast Iterative Graph Computation with Resource Aware Graph Parallel Abstractions. In *ACM Symposium on HPDC*, 179–190.

Zhou, Y.; Liu, L.; Lee, K.; and Zhang, Q. 2015b. Graphtwist: Fast iterative graph computation with two-tier optimizations. *VLDBJ*, 8(11): 1262–1273.

Zhou, Y.; Pu, G.; Ma, X.; Li, X.; and Wu, D. 2021b. Distilled One-Shot Federated Learning. *arXiv preprint arXiv:2009.07999*.

Zhou, Y.; Seshadri, S.; Chiu, L.; and Liu, L. 2014. Graphlens: Mining enterprise storage workloads using graph analytics. In *BigData*.

Zoph, B.; and Le, Q. V. 2017. Neural Architecture Search with Reinforcement Learning. In *ICLR*.